PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGIa
CURSO DE MESTRADO EM INFORMÁTICA

MARCOS ALBERTO MOCHINSKI

PREDIÇÃO DE MULTISSÉRIES TEMPORAIS UNIVARIADAS COM ALGORITMOS DE MINERAÇÃO DE FLUXOS DE DADOS E DEPENDÊNCIA TEMPORAL

CURITIBA

2020

MARCOS ALBERTO MOCHINSKI

PREDIÇÃO DE MULTISSÉRIES TEMPORAIS UNIVARIADAS COM ALGORITMOS DE MINERAÇÃO DE FLUXOS DE DADOS E DEPENDÊNCIA TEMPORAL

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de mestre em Informática.

Orientador: Prof. Dr. FABRÍCIO ENEMBRECK

CURITIBA

2020

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Edilene de Oliveira dos Santos CRB-9/1636

M688p 2020

Mochinski, Marcos Alberto
Predição de multissséries temporais univariadas com algoritmos de mineração de fluxos de dados e dependência temporal / Marcos Alberto Mochinski; orientador, Fabrício Enembreck. -- 2020
222 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2020
Bibliografia: f.171-182

1. Informática. 2. Teoria da predição. 3. Análise de séries temporais. 4. Algoritmos computacionais. 5. Mineração de dados (Computação). I. Enembreck, Fabrício. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título

CDD. 20.ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

DECLARAÇÃO

Declaro para os devidos fins que o aluno **MARCOS ALBERTO MOCHINSKI**, defendeu sua dissertação de Mestrado intitulada **"PREDIÇÃO DE MULTISSÉRIES TEMPORAIS UNIVARIADAS COM ALGORITMOS DE MINERAÇÃO DE FLUXOS DE DADOS E DEPENDÊNCIA TEMPORAL"**, na área de concentração Ciência da Computação, no dia 17 de dezembro de 2020, no qual foi aprovado.

Declaro ainda que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade, firmo a presente declaração.

Curitiba, 02 de fevereiro de 2021.

Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática
Pontifícia Universidade Católica do Paraná

MARCOS ALBERTO MOCHINSKI

PREDIÇÃO DE MULTISSÉRIES TEMPORAIS UNIVARIADAS COM ALGORITMOS DE MINERAÇÃO DE FLUXOS DE DADOS E DEPENDÊNCIA TEMPORAL

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de mestre em Informática.

Trabalho aprovado. CURITIBA, 17 de dezembro de 2020:

Prof. Dr. FABRÍCIO ENEMBRECK
Orientador (PUCPR-PPGIa)

Prof. Dr. JEAN PAUL BARDDAL
(PUCPR-PPGIa)

Prof. Dr. RICHARDSON RIBEIRO
(UFPR)

CURITIBA

Agradecimentos

Agradeço a todos que me deram o incentivo necessário para o desenvolvimento deste trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

"Prediction is very difficult, especially if it's about the future."

NIELS BOHR DOODLE GOOGLE: Great quotes from a man at the nucleus of atomic understanding"by Michael Cavna, www.washingtonpost.com. October 7, 2012.

Resumo

Este estudo apresenta uma abordagem de aprendizagem que faz uso combinado de algoritmos estatísticos e de mineração de fluxos de dados para obter maior acurácia em problemas de predição de multisséries temporais univariadas. Além disso, explora o uso do conceito de dependência temporal no processo de *feature engineering* em busca de um conjunto de características capazes de introduzir no conjunto de dados das séries informações úteis para os processos de treinamento e de predição. Embora alguns algoritmos para séries temporais tenham um desempenho surpreendentemente bom em uma variedade de domínios, é sabido que nenhum é dominante para todos os domínios existentes. Com isso em mente, foi desenvolvida uma técnica baseada na mineração de fluxos de dados e no uso da estratégia de seleção de algoritmos e avaliada a previsão em diferentes conjuntos de dados de séries temporais. Como resultado principal, o estudo apresenta um método denominado AA-ACF que faz uso de auto.arima (estatístico) e AdaGrad (mineração de fluxos de dados) como algoritmos de predição e de coeficientes de autocorrelação (ACF) como base para a criação de atributos. O método proposto procura ampliar o conteúdo existente na literatura no que se refere ao uso de algoritmos de mineração de fluxos de dados na predição de séries temporais, em especial em relação à utilização de tais algoritmos no processamento de séries que possam ser consideradas de curta duração. Após efetuar experimentos com o método na predição de um conjunto com 141.558 séries temporais extraídas de 5 diferentes *datasets* (divididos num total de 19 subconjuntos), foi possível obter um ganho médio positivo de 1,65% em relação ao erro percentual (sMAPE) obtido com o uso isolado do algoritmo auto.arima. Ao analisar os ganhos para os subconjuntos, os ganhos positivos variaram entre 0,079% e 12,429%. Esses ganhos foram obtidos considerando os *forecasts* calculados para todo o horizonte de predição estabelecido para as séries, que consideram predições para 6 a 48 *forecasts* variando de acordo com a periodicidade da série. Ao analisar o ganho médio obtido para *one-step-ahead forecasts* (com horizonte de predição igual a 1) o ganho médio geral passou a ser de 2,081%, com ganhos positivos variando de 0,214% a 32,726% para os diferentes subconjuntos avaliados nos experimentos. Com base nesses resultados, foi possível observar que, para o cenário avaliado, o uso combinado de tecnologias tão diversas possibilitou alcançar maior acurácia no processo de predição, superando os resultados apresentados pelo uso de algoritmos isolados ou outros métodos do estado-da-arte como o algoritmo auto.arima, considerado como *benchmark* na avaliação do método.

Palavras-chaves: predição de séries temporais, algoritmos de mineração de fluxos de dados, multisséries temporais, combinação de algoritmos, dependência temporal.

Abstract

This study presents a learning approach that uses an ensemble of statistical and data stream mining algorithms to obtain better forecast accuracy in multiple univariate time series prediction problems. In addition, it explores the use of temporal dependence in the feature engineering process to establish a set of characteristics that are capable of introducing in the series' data useful information for the training and the prediction processes. Although some multiple time series algorithms perform surprisingly well in a variety of domains, it is well-known that no one is dominant for every existing domain. With that in mind, we developed a technique based on data stream mining and ensemble selection strategy and we evaluated its forecasting in multiple time series datasets. As its main result, the study presents a method named AA-ACF that uses the algorithms auto.arima (statistics) and AdaGrad (data stream mining) and suggests that the combined use of such diverse technologies and the use of autocorrelation coefficients (ACF) as the basis for the creation of additional series attributes can result in improved accuracy. The proposed method aims to expand the existing literature regarding the use of data stream mining algorithms in the prediction of time series, in particular regarding to the use of such algorithms in the processing of series that may be considered of short duration. After experiments with the method in the prediction of a set with 141,558 time series extracted from 5 different datasets (divided into a total of 19 subsets), it was possible to obtain a positive average gain of 1.65% over the sMAPE achieved with the isolated use of the auto.arima algorithm. Evaluating the gains for all the subsets, it was found that the positive gains varied between 0.079% and 12.429%. These gains were obtained considering the forecasts calculated for the original prediction horizons established for the series, which consider predictions from 6 to 48 forecasts varying according to the periodicity of the series. When analyzing the average gain obtained for one-step-ahead forecasts (with a prediction horizon equal to 1) the overall average gain was 2.081%, with positive gains ranging from 0.214% to 32.726% for the different subsets evaluated in the experiments. Based on these results, it was possible to observe that, for the scenario created for the experiments, the combined use of such diverse technologies made it possible to achieve better accuracy in the prediction process, surpassing results presented by the use of isolated algorithms or by the use of state-of-the-art methods such as the auto.arima algorithm, considered the benchmark.

Keywords: time series forecasting, data stream mining algorithms, multiple time series, hybrid ensemble, temporal dependence.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACF	*Autocorrelation Function* (Função de Autocorrelação)
AdaGrad	*Adaptive Gradient Algorithm* (Algoritmo de Gradiente Adaptativo)
ADF	*Augmented Dickey-Fuller test* (Teste de Dickey-Fuller Aumentado)
AIC	*Akaike's Information Criteria*
AMRules	*Adaptive Model Rules*
ANN	*Artificial Neural Network* (Rede Neural Artificial)
AR	Autorregressivo
ARARMA	*ARAR models* (Modelos que fazem análise de esquemas AR ou ARMA para a predição)
ARF-Reg	*Adaptive Random Forest for Data Stream Regression*
ARFF	*Attribute-Relation File Format*
ARIMA	*Autoregressive Integrated Moving Average model* (modelo de Média Móvel Integrada Autorregressiva)
ARMA	Modelo Autorregressivo (AR) e de Média Móvel (MA, *Moving Average*)
BNN	*Bayesian Neural Network* (Rede Neural Bayesiana)
CART	*Classification & Regression Trees* (Árvores de Classificação e Regressão)
CSV	*Comma-separated Values*
DSM	*Data Stream Mining* (Mineração de Fluxos de Dados)
ETS	*Exponential Smoothing State Space model* (modelo de Espaço de Estado de Suavização Exponencial)
FIMT-DD	*Fast Incremental Model Tree with Drift Detection*
GP	*Gaussian Processes* (Processos Gaussianos)
HMM	*Hidden Markov Models* (Modelos Ocultos de Markov)
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)

IoT	*Internet of Things* (Internet das Coisas)
KPSS	*Kwiatkowski-Phillips-Schmidt-Shin test* (KPSS test)
LOESS	*Locally Estimated Scatterplot Smoothing*
M3	*M3-Competition* (Competição M3 de predição de séries temporais)
M4	*M4-Competition* (Competição M4 de predição de séries temporais)
MA	*Moving Average* (Média Móvel)
MAPE	*Mean Absolute Percentage Error*
MLP	*Multilayer Perceptron*
MOA	*Massive Online Analysis*
MOV-AVG	*Moving Average* (Média Móvel)
MR	*Multiple Regression* (Regressão Múltipla)
NNETAR	Modelo de Rede Neural para predição de Séries Temporais
ORTO	*Online Option Trees for Regression*
PACF	*Partial Autocorrelation Function plot* (Gráfico de Função de Autocorrelação Parcial)
PCA	*Principal Component Analysis* (Análise Componentes Principais)
R	Linguagem de Programação R
RBF	*Radial Basis Function Neural Network* (Rede Neural de Função de Base Radial)
RMSE	*Root Mean Squared Error*
RNN	*Recurrent Neural Network* (Rede Neural Recorrente)
RW-DRIFT	*Random Walk with Drift*
SARIMA	*Seasonal Autoregressive Integrated Moving Average* (Média Móvel Integrada Autorregressiva Sazonal)
SES	*Simple Exponential Smoothing* (Suavização Exponencial Simples)
SGD	*Stochastic Gradient Descent*
sMAPE	*Symmetric Mean Absolute Percentage Error*

SSA	*Singular Spectrum Analysis* (Análise de Espectro Singular)
STLM-AR	*Seasonal and Trend decomposition using Loess with AR modeling*
SVM	*Support Vector Machine* (Máquina de Vetor de Suporte)
SVR	*Support Vector Regression* (Regressão de Vetor de Suporte)
TBATS	*Exponential Smoothing State Space Trigonometric, Box-Cox transformation, ARMA errors, Trend and Seasonal components model*
VM	*Virtual Machine* (Máquina Virtual)

Sumário

1 Introdução

De acordo com a versão *online* do dicionário Michaelis [1], a palavra "predição" tem os seguintes significados: *1. Ato ou efeito de predizer ou de afirmar o que se acredita que vai acontecer no futuro; prognóstico, vaticínio; 2. Aquilo que se prediz.*

Ou seja, predizer significa fazer um prognóstico, estimar, prever, tentar antever, com base nas habilidades e recursos disponíveis, o que o futuro reserva em relação a algo de interesse. Pode ser que se tenha intenção de prever a chance de realização de um projeto pessoal, ou então, simplesmente tentar antever a possibilidade de um evento cotidiano acontecer, como o de prever qual a chance de seu time de futebol preferido chegar à final do campeonato.

A predição pode ser entendida também como um evento numérico e é nessa definição que este estudo está interessado. No caso, o interesse está em estudar o uso de algoritmos estatísticos e algoritmos computacionais na predição de eventos futuros em séries temporais.

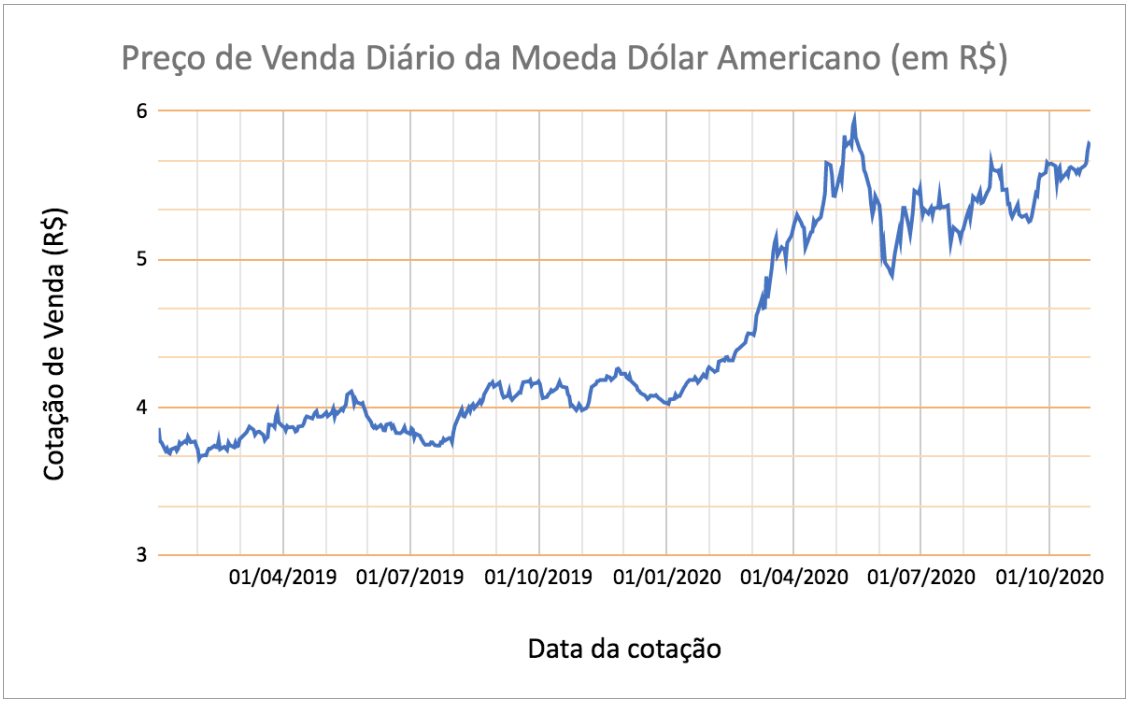
As séries temporais estão presentes no nosso dia-a-dia nos indicadores econômicos que são apresentados nos jornais, nos gráficos que mostram a variação de vendas de um determinado produto, nos dados observados de um eletroencefalograma, na oscilação da cotação de uma ação da bolsa de valores, nos dados de um censo que mostre a variação populacional de um município, enfim, podem ser encontrados nos mais diferentes domínios de nossa sociedade, em diferentes áreas da ciência (HANNAN, 1970). A Figura 1 apresenta um exemplo de série temporal diária com as cotações de venda da moeda Dólar Americano.

Os cenários de aplicação de séries temporais são os mais diversos e as informações registradas para uma série podem variar de acordo com o sistema em que é armazenada ou o tipo de informação que representa. Para algumas séries, podem-se ter poucos dados, resultados da anotação de valores de eventos que podem ocorrer, por exemplo, uma vez por ano. Para outras, por sua vez, pode ser que se tenha interesse em identificar a variação de informação a cada milisegundo e, dessa forma, resultar em um grande conjunto de dados a serem armazenados.

O fato é que, independente da natureza da série e do conjunto de dados que a representam, técnicas de predição de eventos futuros são cada vez mais necessárias, em especial devido ao interesse cada vez maior da sociedade em extrair informação relevante a partir de dados obtidos pelas tecnologias modernas. Na era de conceitos como *Big Data* e *IoT* (*Internet of Things*), o volume de dados e a velocidade com que devem ser processados é cada vez maior e é necessário fazer uso de técnicas inovadoras de forma a transformar esses repositórios em informação útil (MORALES et al., 2016).

[1] http://michaelis.uol.com.br/busca?r=0&f=0&t=0&palavra=predi%C3%A7%C3%A3o

Figura 1 – Exemplo de série temporal que apresenta a cotação de venda diária da moeda Dólar Americano (USD) para o período de 01/01/2019 a 31/10/2020. Fonte dos dados: Banco Central do Brasil (bcb.gov.br), plataforma Olinda, acesso em 07/11/2020.



Fonte: Autoria própria.

No caso da predição, os métodos estatísticos e o uso de tecnologias mais recentes como as redes neurais e técnicas de *machine learning* (aprendizagem de máquina) estão mais presentes nas soluções utilizadas no dia-a-dia, mesmo que sejam em competições aplicadas a essa área, ou em soluções criadas para tratar de forma personalizada um determinado problema. Isso pode ser evidenciado nos resultados de competições de predições de séries temporais como a M3 (MAKRIDAKIS; HIBON, 2000), a M4 (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a) ou a M5 (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2020), entre outras, em que novas técnicas ou combinações de técnicas se destacam na busca por processos de predição cada vez mais precisos.

Este estudo, então, tem como objetivo incluir nesse universo de soluções de predição um novo método que faça uso de técnicas de *data stream mining* (mineração de fluxos de dados). Considere, por exemplo, que uma estação meteorológica faz medições de temperatura de uma determinada localização de maneira contínua e em intervalos de tempo pré-definidos. Tem-se, nesse caso, um exemplo de fluxo contínuo de dados que pode ser analisado por técnicas de *data stream mining*. Este estudo, por sua vez, procura fazer a análise de algoritmos de predição em um cenário que não fique limitado à utilização da

técnica a uma série isolada, mas sim a um conjunto de várias séries, um cenário denominado de multisséries temporais.

É importante destacar que, de forma geral, algoritmos de *machine learning* demandam uma grande quantidade de dados para treinamento e criação do modelo representativo de um dado problema. Um desafio importante, então, é o de verificar a aplicabilidade de algoritmos de *data stream mining* no processamento de conjuntos de séries temporais que possam ter séries com um número de observações bem reduzido e, mesmo assim, apresentar nível de acurácia competitivo com os obtidos pelo uso de técnicas do estado-da-arte.

Dessa forma, a questão de pesquisa do trabalho corresponde a identificar qual a aplicabilidade de algoritmos de *data stream mining* para a predição (*forecasting*) de multisséries temporais.

O estudo é motivado pela existência de cenários com elevado número de séries temporais (com poucas ou muitas observações), que poderiam se beneficiar pelo uso de *data stream mining*. A literatura existente, apesar de apresentar muitas referências relacionadas a séries temporais, carece de conteúdos sobre algoritmos para multisséries temporais, em especial que façam uso de abordagens incrementais e com o uso de algoritmos de *data stream mining*. Além de procurar reduzir essa lacuna, o estudo avalia o uso isolado e combinado de algoritmos de *data stream mining* e algoritmos estatísticos pela relevância que algoritmos estatísticos apresentam na área de predição de séries temporais. Em relação aos algoritmos de séries temporais, são comuns os estudos sobre o uso de técnicas de aprendizagem de máquina, modelos estatísticos e técnicas híbridas para a predição de séries dessa natureza. Mas a aplicação de algoritmos de mineração fluxos de dados em problemas de séries temporais é pouco observada. Os autores em (BOULEGANE; BIFET; MADHUSUDAN, 2019), que apresentam um estudo sobre o uso de *ensemble* (combinação de algoritmos) para a predição de séries temporais em fluxos de dados, citam que a mineração de dados dessa natureza é desafiadora devido à não-estacionariedade das séries, à existência de diferentes regimes, às mudanças de conceito e à ocorrência de conceitos recorrentes. Indicam, também, que o uso de métodos de *ensemble* são extensivamente estudados para tarefas de classificação, mas menos explorados para tarefas de predição e de regressão.

Além disso, é de grande relevância para o estudo explorar o uso do conceito de dependência temporal como base para o processo de *feature engineering* de forma a adicionar ao conjunto de dados originais das séries atributos capazes de representar informações históricas das séries que auxiliem no processo de treinamento e predição.

Em relação às contribuições que o estudo pode promover destacam-se a ampliação do universo de aplicação de algoritmos de *data stream mining* para a área de predição de multisséries temporais, a proposição de modelo que ofereça suporte, inclusive, ao processamento de séries com baixo número de observações e evidenciar a possibilidade de

uso combinado de algoritmos estatísticos e algoritmos de mineração de fluxos de dados.

1.1 Objetivos

1.1.1 Objetivo Principal

Este estudo tem como objetivo principal desenvolver um método orientado a *data stream mining* capaz de alcançar resultados de predição de multisséries temporais competitivos em relação ao estado da arte, se beneficiando das características de algoritmos de *data stream mining*.

Sob o ponto de vista de hipótese de pesquisa, o estudo procura verificar se a combinação de algoritmos estatísticos com algoritmos de *data stream mining* possibilita melhor acurácia no processo de *forecasting* de multisséries temporais univariadas, em especial em comparação ao uso isolado de tais algoritmos.

1.1.2 Objetivos Específicos

Este estudo tem como objetivos específicos analisar a performance de algoritmos de *data stream mining* no contexto de multisséries temporais, experimentar técnicas para a identificação de conjuntos de características relevantes para o processo de predição (em especial atributos que consigam expressar a dependência temporal existente nas séries), estudar diferentes formas de representação de séries temporais, propor um método de predição de séries temporais que faça uso combinado de algoritmos de mineração de fluxos de dados e de algoritmo estatístico e comparar resultados com os apresentados por algoritmos da literatura.

2 Revisão da Literatura

Com o objetivo de propor um método para a predição de multisséries temporais univariadas fazendo uso de algoritmos de mineração de fluxos de dados, é importante conhecer conceitos associados aos temas envolvidos e ter uma noção atualizada acerca da literatura existente, a terminologia e os conceitos mais relevantes.

Este capítulo apresenta conceitos, referências e algumas considerações de interesse sobre os temas séries temporais[1], multisséries temporais, mineração de fluxos de dados e dependência temporal, que representam as principais áreas associadas a este estudo.

2.1 Séries Temporais

Considere o seguinte cenário: o departamento comercial de uma empresa registra, todos os meses, as vendas de seu principal produto em um sistema e acompanha em um gráfico a variação no volume de vendas. Certamente, é de interesse do diretor comercial dessa empresa saber, de antemão, se a tendência das vendas nos próximos meses é de aumento ou de diminuição, pois isso pode afetar diretamente o tamanho da equipe de vendas que deve ser colocada em campo e, também, por essa informação ter impacto direto no volume de itens a serem montados pela equipe da produção.

Ou então, numa outra situação, o responsável pela gestão de um *datacenter* tem percebido que o aumento no número de máquinas virtuais (*virtual machines*, VMs) alocadas por seus clientes tem resultado num aumento no consumo de energia de forma não proporcional, talvez porque os processamentos realizados nesses ambientes estejam cada vez mais complexos, demandando mais energia. Com isso, é interessante que esse gestor tenha condições de prever, com certa faixa de segurança, qual o volume máximo de VMs que poderá disponibilizar de forma que a rede elétrica existente não precise passar por ampliação pelo menos pelo período de um ano (para quando estão previstos novos investimentos em infraestrutura).

Os cenários descritos acima mostram exemplos de situações que podem ser representadas pelo uso de séries temporais. Elas estão presentes no dia-a-dia das mais diferentes áreas, podendo representar informações de diversas categorias, como por exemplo: informações demográficas, cotações do mercado financeiro, dados sobre a macroeconomia, informações médicas, dados educacionais, crescimento de vendas em determinado setor, entre outros.

[1] Na seção referente a séries temporais, algumas equações podem apresentar notação diferente da encontrada em suas fontes originais, devido a adaptações feitas com o objetivo de padronização de notação entre as diversas fontes consultadas.

Uma série temporal pode ser definida como uma sequência ordenada de variáveis de valores reais observadas a intervalos equidistantes (Equação 2.1), (ESLING; AGON, 2012) (ADHIKARI; AGRAWAL, 2013) (MAKRIDAKIS, 1976).

$$X_t = (X_1, X_2, X_3, \ldots, X_n), X_i \in \mathbb{R} \tag{2.1}$$

Kolmogoroff (1933 apud MAKRIDAKIS, 1976) mostrou que uma série temporal é equivalente a uma distribuição de probabilidades em um número n de dimensões em que cada observação corresponde a uma dimensão, que engloba o conjunto de observações identificados pelo processo real (X_t') e um termo que representa um ruído branco (*white noise term*), e_t. A representação de acordo com essa definição é indicada na Equação 2.2:

$$X_t = X_t' + e_t \tag{2.2}$$

onde X_t é expresso como um desvio da média geral da série. Makridakis (1976) comenta que, em 1807, Joseph Fourier mostrou que qualquer série temporal pode ser expressa como uma soma de senos e cossenos. No entanto, Bartlett (1935 apud MAKRIDAKIS, 1976) destaca que a maior dificuldade no uso dessa definição é que ela não consegue tratar a existência de ruídos na série e, com isso, inviabiliza seu uso com propósito de predição. Brillinger (2002), por sua vez, faz uma compilação interessante do trabalho desenvolvido por John W. Tukey apresentando uma cronologia de seus estudos sobre o tema séries temporais, em especial em relação à área de análise espectral e no desenvolvimento de um método de computação rápida da transformada de Fourier (*Cooley-Tukey FFT*, *Fast Fourier Transform*, (COOLEY; TUKEY, 1965)).

De acordo com (MAKRIDAKIS, 1976), o uso de técnicas baseadas em séries temporais permite separar o ruído (ou aleatoriedade) do padrão (comportamento) de um processo real (sistema). E isso pode ser feito pelo uso de técnicas de análise e pelo uso de métodos aplicados a séries temporais. Os autores em (ADHIKARI; AGRAWAL, 2013) sugerem que uma série temporal é não-determinística por natureza e isso indica que não podemos prever, com certeza, os eventos futuros de uma série. Ou seja, a sequência de observações de uma série é resultado do processo estocástico que a produz.

Esling e Agon (2012) complementam que uma série temporal pode ser definida como um conjunto de instantes de tempo contíguos e que uma série pode ser univariada, ou multivariada quando várias séries abrangem, simultaneamente, várias dimensões no mesmo intervalo de tempo. De uma forma geral, pode-se dizer que uma série univariada é aquela cujas observações se referem a uma única variável. Uma série multivariada, por sua vez, é aquela que contém informações relativas a mais de uma variável.

Makridakis (1976) faz uso de uma abordagem interessante para discorrer sobre séries temporais e, em resumo, sugere que o estudo de séries temporais pode ser dividido

em duas partes:

- Análise de séries temporais, que se subdivide em:
  - Autocorrelação, em que a análise ocorre no domínio do tempo.
  - Métodos espectrais, em que a análise ocorre no domínio da frequência.
- Métodos de séries temporais, que estão organizados em:
  - Autorregressão / Médias móveis.
  - Função de transferência.
  - Filtragem.
  - Suavização exponencial.
  - Decomposição.

Sugere ainda que, de acordo com a função, os métodos estão organizados em:

- Métodos para predição (*forecasting*):
  - Modelos autorregressivos / médias móveis.
  - Modelos de suavização exponencial.
  - Métodos de decomposição.
- Métodos de controle:
  - Modelos de função de transferência.
- Métodos para estimar o estado atual de um processo real (mais aplicados à área de Engenharia, como na análise de circuitos elétricos, em que os dados de entrada devem estar entre determinadas faixas de valores):
  - Métodos de filtragem.

Do ponto de vista deste estudo, o interesse será direcionado à análise de métodos para predição (*forecasting*). De acordo com (HYNDMAN; ATHANASOPOULOS, 2018), a previsibilidade de um evento depende de fatores como: o quanto compreendemos sobre os fatores que contribuem para a ocorrência de tal evento; a quantidade de dados que temos à disposição para a análise e a influência que a predição realizada sobre um evento afeta a predição desse próprio evento. Comentam ainda que boas predições capturam padrões genuínos e os relacionamentos que existem nos dados históricos, mas sem reproduzir aqueles eventos passados que não ocorreriam novamente de forma natural.

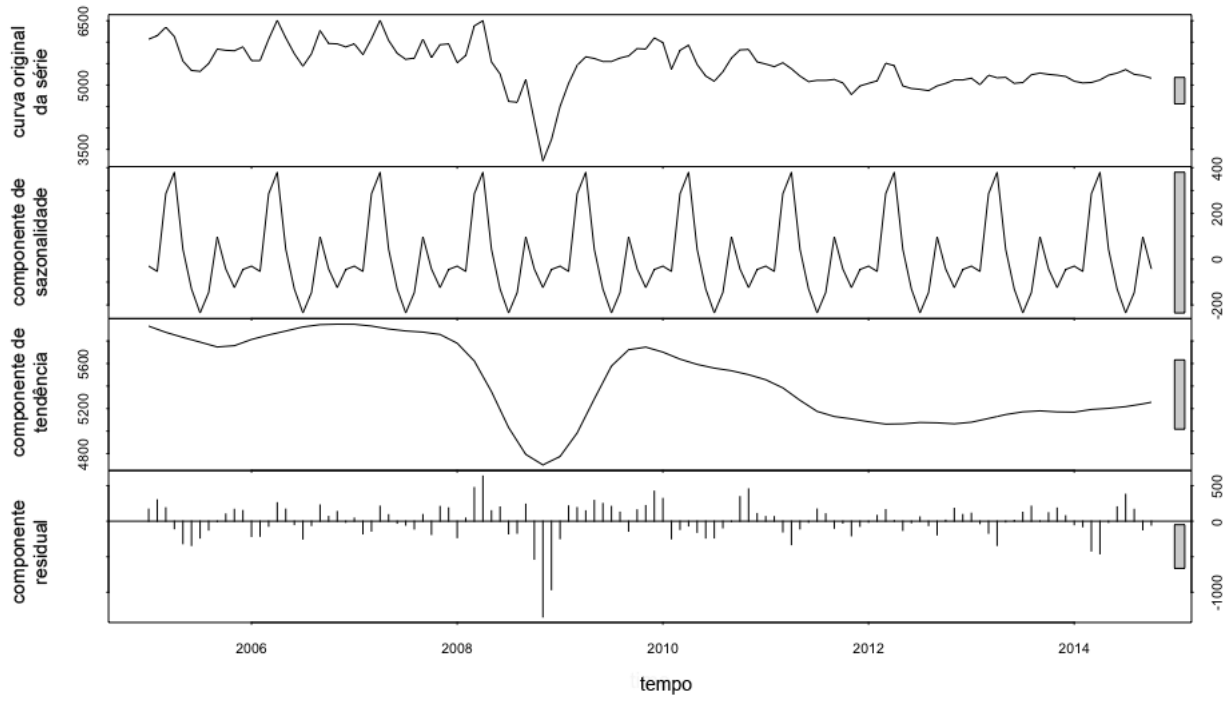
Para isso, com o intuito de manter o foco no tema predição, inicia-se a explicação falando sobre os componentes de uma série temporal, uma vez que as características inerentes de cada série podem interferir na sua análise e no método melhor aplicável ao seu estudo.

Ao realizar a decomposição de uma série temporal, os seguintes componentes podem ser observados (conforme (ADHIKARI; AGRAWAL, 2013), (MAKRIDAKIS, 1976) e (HYNDMAN; ATHANASOPOULOS, 2018)):

- Componente de tendência (*trend component, T*) - indica se a tendência das observações de uma série é de crescimento, diminuição ou estagnação.
- Componente de sazonalidade (*seasonal component, S*) - indica a presença de variações sazonais nas observações da série temporal, normalmente associadas a aspectos culturais (como eventos que ocorrem anualmente no mesmo período do ano, ou nos mesmos dias de semana, como a folga aos domingos, por exemplo), ou a alterações de comportamento que afetem a geração de eventos da série (como o aumento de vendas de bebidas quentes no inverno, por exemplo).
- Componente cíclico (*cyclical component, C*) - corresponde a comportamentos dos dados da série que se repetem, mas mais a médio e longo prazo, representando uma alteração nas observações associadas, por exemplo, a um ciclo de produto, ou a um ciclo de negócio. O aumento de vendas de um modelo de carro ocorrendo sempre que se lança um novo modelo pode indicar a ocorrência de um ciclo na série temporal de vendas, que ocorre sempre em decorrência do lançamento do novo modelo.
- Componente irregular, ou componente aleatório (*random component, R*) - representa, geralmente, fatores que não podem ser previstos e que afetam as observações que compõem a série. Esse componente também pode ser chamado de componente residual da série e compreende todas as variações que não são compreendidas pela curva ideal da série, expressa por seus demais componentes (tendência, sazonalidade, cíclico). No mundo real, compreende todas as variações observadas ao avaliar os valores de uma série. Por mais que as séries tenham um comportamento esperado dentro de certos limites, as variações residuais são percebidas e por isso devem estar previstas como um componente aleatório da série. Por exemplo: considerando que o volume esperado de vendas para um determinado mês seja de 200 unidades, um aumento no volume de vendas para 205 unidades pode indicar um crescimento não esperado (aleatório) de vendas de 5 unidades.

A Figura 2 apresenta um exemplo de gráfico com os principais componentes de uma série temporal (no caso, a série M48000 de M4 (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a)), em que estão indicadas a curva original da série (*data*),

Figura 2 – Componentes da decomposição da série M48000 de M4 (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a). Gráfico gerado com o uso da função *stl* do pacote *stats* na linguagem R (R Core Team, 2018)



Fonte: Autoria própria

a sua componente de sazonalidade (*seasonal*), a sua componente de tendência (*trend*), e sua componente residual (*remainder* ou componente aleatória (*random component*)). Na Figura 2 a decomposição é feita com o uso da função *stl* (*Seasonal Decomposition of Time Series by Loess*) do pacote *stats* (R Core Team, 2018). De acordo com a documentação dessa função, a componente sazonal é obtida pela aplicação de uma suavização com o método LOESS (*Locally Estimated Scatterplot Smoothing*, denominado também de método de regressão local por Jacoby (2000)) sobre as sub-séries sazonais (por exemplo, considerando as séries compostas pelos mesmos meses em diferentes anos). Os valores sazonais são removidos da série e o restante dos dados são suavizados para encontrar a tendência. O nível geral é removido da componente sazonal e adicionada à componente de tendência. O processo é iterativo e repetido algumas vezes. A componente residual (ou componente aleatória) corresponde aos valores restantes da série depois de retirados os valores referentes a sazonalidade e tendência.

Outro conceito importante relacionado a uma série temporal é o conceito de estacionariedade, que indica se uma série é estacionária ou não-estacionária. De acordo

com (HYNDMAN; ATHANASOPOULOS, 2018) uma série estacionária é aquela cujas propriedades não dependem do tempo em que a série é observada. Então, séries temporais com tendência, ou que apresentem sazonalidade, não se encaixam nessa categoria e passam a ser denominadas de séries não-estacionárias uma vez que tendência e sazonalidade vão interferir no valor das observações da série em diferentes momentos do tempo. Conforme (MAKRIDAKIS, 1976), uma série temporal é estacionária se os valores de suas observações variam em torno de uma média constante e suas propriedades estatísticas são independentes do tempo.

Em relação aos métodos de predição, em especial sobre os modelos estatísticos, seguem considerações sobre os principais modelos.

2.1.1 Modelos Autorregressivos / Médias Móveis

De acordo com (ADHIKARI; AGRAWAL, 2013) um modelo ARMA(p,q) é uma combinação de um modelo Autorregressivo (AR(p)) e de um modelo de Médias móveis (MA(q)), apropriado para a modelagem de séries temporais univariadas.

Hyndman e Athanasopoulos (2018) explicam que, num modelo de Autorregressão (AR), a predição de uma variável de interesse é feita usando uma combinação linear de valores passados dessa variável.

Adhikari e Agrawal (2013) trazem uma explicação um pouco mais completa ao indicar que, num modelo AR(p) (modelo Autorregressivo de ordem p), o valor futuro de uma variável é uma combinação linear dos valores de observações passadas e um erro aleatório somados a uma constante. O erro aleatório incluído na definição serve certamente para compensar os ruídos existentes na série.

De acordo com (JAIN, 2017) modelos AR são autorregressivos porque a regressão da variável dependente da série é feita a partir das variáveis da série. Cita o exemplo de que as vendas de um período podem ser calculadas a partir da regressão das vendas de um ou mais períodos anteriores e sugere a Equação 2.3 para a representação de um modelo AR de ordem p:

$$Y_t = C + B_1\, Y_{t-1} + B_2\, Y_{t-2} + ... + B_p\, Y_{t-p} + e_t \tag{2.3}$$

onde:

Y_t: Variável dependente.

C: Constante.

$Y_{t-1}, Y_{t-2}, ...Y_{t-p}$: Variáveis independentes (valores atrasados, *lagged*, da variável).

$B_1, B_2, ...B_p$: Coeficientes de regressão.

e_t: Termo correspondente ao erro (ruído branco, *white noise*) representando o efeito de eventos aleatórios sobre a variável dependente.

Em relação aos modelos de médias móveis (MA, *moving average models*), Jain (2017) explica que eles assumem que a variável dependente Y_t depende de erros passados (residuais). Adhikari e Agrawal (2013) informam que um modelo de média móvel é uma regressão linear da observação atual da série temporal contra os erros aleatórios de uma ou mais observações anteriores, ou seja, um modelo MA(q) usa erros passados como variáveis explanatórias. Jain (2017) apresenta a seguinte representação matemática (Equação 2.4) para um modelo MA de ordem q:

$$Y_t = C + W_1\, e_{t-1} + W_2\, e_{t-2} + ... + W_q e_{t-q} + e_t \tag{2.4}$$

onde:

Y_t: Variável dependente.

C: Constante.

$W_1, W_2, ...W_q$: Pesos atribuídos.

e_t: Termo correspondente ao erro (*residual error*) do período t.

$e_{t-1}, e_{t-2}, ...e_{t-q}$: Valor do erro dos períodos anteriores $t-1$, $t-2$, e assim por diante.

q: Número de termos.

Um modelo ARMA(p,q) pode ser representado então como (Equação 2.5):

$$Y_t = C + B_1\, Y_{t-1} + B_2\, Y_{t-2} + ... + B_p\, Y_{t-p} + e_t + W_1\, e_{t-1} + W_2\, e_{t-2} + ... + W_q e_{t-q} \tag{2.5}$$

De acordo com (ADHIKARI; AGRAWAL, 2013) modelos ARMA são aplicáveis a séries estacionárias. Os autores indicam que séries temporais que contêm tendência e sazonalidade são não-estacionárias por natureza. Então, para os casos em que o modelo ARMA não se aplica, sugere a utilização do modelo denominado ARIMA(p,d,q) (acrônimo de *Autoregressive Integrated Moving Average model*) que é capaz de transformar uma série não estacionária em estacionária pela aplicação de diferenciação nos dados da série. Em ARIMA(p,d,q), "p" corresponde ao coeficiente de autorregressão, "d" indica o grau de diferenciação e "q" o coeficiente de médias móveis.

O autor em (PRABHAKARAN, 2019) sugere que o primeiro passo para uso do modelo ARIMA é tornar a série estacionária. Para isso, a série deve passar pela aplicação de (pelo menos) uma diferenciação nos dados. Após isso, o uso de um teste de *Dickey-Fuller* aumentado (ADF, *Augmented Dickey-Fuller test,* (DICKEY; FULLER, 1979)) pode indicar se a série é estacionária ou não. Para (HYNDMAN; ATHANASOPOULOS, 2018), outra opção é a de avaliar a estacionariedade de uma série pela geração de um gráfico ACF (ACF *plot*, ou *Autocorrelation Function plot*), pois, em séries estacionárias, os valores no gráfico ACF decrescem relativamente rápido para zero, enquanto que, em séries não estacionárias os dados decrescem mais lentamente.

Com a série transformada em estacionária, o próximo passo é calcular os valores para os componentes "p", correspondente ao fator AR (autorregressão), e "q", correspondente ao fator MA (médias móveis), de forma a poder ser resolvido por um modelo ARMA. Para isso, os autores em (PRABHAKARAN, 2019) e (HYNDMAN; ATHANASOPOULOS, 2018), sugerem, entre outras possibilidades, o uso combinado de gráficos de autocorrelação (ACF) e autocorrelação parcial (PACF). Conforme (HYNDMAN; ATHANASOPOULOS, 2018), a autocorrelação parcial mede a relação entre as observações da série (por exemplo: y_t e y_{t-k}) após remover o efeito dos *lags* $1, 2, 3, ...k-1$. Na seção Dependência Temporal deste Capítulo de Revisão de Literatura, os termos autocorrelação e autocorrelação parcial são descritos com mais detalhes.

A princípio, o modelo ARIMA suporta séries que não apresentam sazonalidade. Mas, variações desse modelo como o SARIMA (*Seasonal Autoregressive Integrated Moving Average*, SARIMA$(p,d,q)(P,D,Q)^s$), (JAIN, 2017) (ADHIKARI; AGRAWAL, 2013), oferecem suporte a séries não-estacionárias, com sazonalidade. Na representação de SARIMA, os parâmetros (p,d,q) correspondem aos seus componentes não sazonais (tal como num modelo ARIMA comum) e $(P,D,Q)^s$ aos seus componentes sazonais, em que P corresponde ao seu fator AR (autorregressão) sazonal, D ao seu fator I (diferenciação) sazonal e Q ao seu fator MA (média móvel) sazonal. O termo s em $(P,D,Q)^s$ indica simplesmente quantos períodos compõem a sazonalidade da série, e não atua como um expoente. Se for um período de 12 meses, s será igual a 12. A notação SARIMA$(p,d,q)(P,D,Q)_s$ é, então, equivalente.

Do ponto de vista histórico, Makridakis (1976) informa que G.E.P. Box e G.M. Jenkins tiveram uma grande participação em tornar os modelos ARMA mais aceitos pelas comunidades de estatísticos e da engenharia, especialmente por suas contribuições teóricas sobre o tema. Por essa razão, os nomes Box-Jenkins podem ser vistos como sinônimos de modelos autorregressivos e de médias móveis, certamente por terem estendido a proposta original do modelo ARMA (introduzido por Yule (1927 apud MAKRIDAKIS, 1976), Walker (1931 apud MAKRIDAKIS, 1976) e Slutzky (1937 apud MAKRIDAKIS, 1976) para o esquema denominado *Autoregressive Integrated Moving Average*.

Gooijer e Hyndman (2006) comentam que a publicação de Box e Jenkins (1976 apud GOOIJER; HYNDMAN, 2006) integrou o conhecimento existente na época. Além disso, informam que os autores (Box e Jenkins) desenvolveram um ciclo iterativo e coerente em três estágios para a identificação da série temporal, a estimativa e a verificação, conhecida como abordagem Box-Jenkins.

2.1.1.1 Algoritmo auto.arima

O algoritmo ARIMA foi selecionado para ser o algoritmo estatístico de predição a atuar de forma combinada com o algoritmo de *data stream mining* AdaGrad (DUCHI; HAZAN; SINGER, 2011) no método proposto por este estudo. Em experimentos iniciais, além de ser utilizado na predição de séries temporais, foi avaliado, também, no processo de *feature engineering*. Mais detalhes sobre esses experimentos podem ser encontrados no Apêndice A.

Nesta seção são apresentados detalhes sobre a implementação de ARIMA selecionada para este estudo e para uso no método AA-ACF, que corresponde à função *auto.arima()* do pacote *forecast* para a linguagem R, denotado como *forecast::auto.arima* ou simplesmente auto.arima.

De acordo com (HYNDMAN; ATHANASOPOULOS, 2018), auto.arima é uma função em R que usa uma variação do algoritmo de Hyndman-Khandakar (HYNDMAN; KHANDAKAR, 2008), que seleciona de forma automática os parâmetros mais adequados para ARIMA. A referência explica também que, o AIC (*Akaike's Information Criteria*), é um indicador útil na definição de parâmetros de preditores de regressão e, dessa forma, útil na definição da ordem de um modelo ARIMA. Para o algoritmo de Hyndman-Khandakar é utilizado o AICc que corresponde a um valor corrigido de AIC para os parâmetros de ARIMA e usado especialmente na definição dos parâmetros p e q.

O Algoritmo 1, algoritmo de Hyndman-Khandakar, apresentado a seguir e extraído de (HYNDMAN; ATHANASOPOULOS, 2018), explica de que forma os parâmetros de ARIMA são definidos para a função auto.arima.

Algoritmo 1 - Algoritmo Hyndman-Khandakar para modelagem automática de ARIMA, conforme descrito em (HYNDMAN; ATHANASOPOULOS, 2018):

---

1. *O parâmetro "d", referente à ordem de diferenciação, é um número que varia de 0 a 2 (0<=d<=2) e é determinado usando repetidos testes KPSS (KWIATKOWSKI et*

al., 1992). O Kwiatkowski-Phillips-Schmidt-Shin (KPSS test) testa se uma série é estacionária ou não e é um modelo de teste de raiz unitária (unit root test).

2. *Os parâmetros "p" e "q" são escolhidos a partir do menor valor de AICc obtido após diferenciar os dados d vezes. O modelo usa uma estratégia de pesquisa gradual (stepwise search) para percorrer o modelo de espaço de estados (model space) para definir os valores de p e q:*

   a) *Inicialmente, 4 modelos são avaliados:*

      - *ARIMA(0,d,0)*
      - *ARIMA(2,d,2)*
      - *ARIMA(1,d,0)*
      - *ARIMA(0,d,1)*
      - *Uma constante é incluída, a menos que d seja d=2. Se d<=1, então mais um modelo é considerado: ARIMA(0,d,0), sem uma constante*

   b) *O modelo com menor AICc identificado no passo 2(a) é setado como o "modelo atual".*

   c) *São efetuadas variações no modelo atual, conforme abaixo:*

      - *Variação de p e/ou q em ± 1*
      - *Incluir/excluir a constante c do modelo atual.*
      - *O melhor modelo encontrado passa a ser o modelo atual.*

   d) *Repete-se o passo 2(c) até que nenhum valor menor de AICc seja obtido.*

A escolha da função auto.arima para este trabalho foi motivada pelo fato de ela ser uma implementação automática de ARIMA. Para um problema de análise de multisséries temporais, essa parece ser uma abordagem interessante, uma vez que o método seleciona automaticamente os parâmetros de ARIMA para a série de acordo com as regras do algoritmo, podendo sugerir também modelos sazonais de ARIMA (SARIMA). Em contrapartida, esse tipo de algoritmo acarreta num aumento do custo computacional se comparado ao custo do uso do método ARIMA com parâmetros previamente estabelecidos.

2.1.2 Modelos de Suavização Exponencial

De acordo com (HYNDMAN; ATHANASOPOULOS, 2018), os métodos de suavização exponencial utilizam médias ponderadas de observações passadas de uma série, nas quais os pesos decaem exponencialmente à medida que as observações vão ficando mais distantes. Com isso, quanto mais recente a observação de uma série, maior o peso atribuído a ela no processo de predição.

Em (GOOIJER; HYNDMAN, 2006) os autores comentam sobre uma taxonomia criada por Hyndman em 2002 para os métodos de suavização exponencial, que se baseia no tipo de tendência apresentada pela série (nenhuma, aditiva, aditiva com amortecimento (*damped additive*), multiplicativa e multiplicativa com amortecimento) e também na sazonalidade da série (nenhuma, aditiva, multiplicativa). Com isso, sugerem 15 diferentes métodos, em que os mais conhecidos seriam o método SES (para séries sem tendência, nem sazonalidade), o método linear de Holt (para séries com tendência aditiva e sazonalidade aditiva) e o método multiplicativo de Holt-Winters (para séries com tendência aditiva e sazonalidade multiplicativa).

Para Jain (2017), no modelo SES (*Simple exponential smoothing*, modelo de suavização exponencial simples) existem duas propriedades básicas: nesse tipo de modelo é dado um peso maior aos dados mais recentes e menor peso aos demais; além disso, o modelo automaticamente se ajusta para o erro experimentado durante o período atual. O modelo SES pode ser expresso como na Equação 2.6:

$$\hat{X}_{t+1} = \alpha X_t + (1 - \alpha)\hat{X}_t \tag{2.6}$$

onde:

$\hat{X}_{t+1}$: Predição (*forecast*) do próximo período.

α: Constante de suavização.

X_t: Valor do período corrente.

$\hat{X}_t$: Predição (suavizada) do período corrente.

2.1.3 Métodos de Decomposição

Os métodos de decomposição fazem uso dos componentes de uma série (tendência, T; sazonalidade, S; componente cíclico, C; e componente aleatório, R) para a predição.

Esses componentes podem ser analisados individualmente ou em conjunto e auxiliar no cálculo de *forecasts* de um determinado produto. Analisando a sazonalidade é possível, por exemplo, identificar a variação de vendas de um mês para outro e, dessa forma, conseguir estimar valores futuros. Isso pode ser feito pelo cálculo do Índice de Sazonalidade Específico (JAIN, 2017). Tendo-se a variação de sazonalidade, é possível fazer um uso combinado da análise de tendência e conseguir uma estimativa de crescimento ou diminuição de vendas, por exemplo, que leve em consideração esses dois componentes (no caso, sazonalidade e tendência).

Uma vez estimados individualmente os valores futuros dos componentes de uma série é possível, por exemplo, fazer o cálculo de predição usando as seguintes relações

existentes entre os elementos de uma série temporal X_t ((ADHIKARI; AGRAWAL, 2013) e (MAKRIDAKIS, 1976)):

- Modelo multiplicativo (Equação 2.7):
  - Nesse modelo, os componentes não são necessariamente independentes e podem afetar uns aos outros.

$$X_t = S_t \times T_t \times C_t \times R_t \tag{2.7}$$

- Modelo aditivo (Equação 2.8):
  - Nesse modelo, os componentes são independentes entre si.

$$X_t = S_t + T_t + C_t + R_t \tag{2.8}$$

Levando-se em consideração todos os componentes, certamente é possível alcançar estimativas mais precisas uma vez que diferentes variáveis do processo são levadas em consideração.

2.1.4 Outros Métodos

Hyndman e Athanasopoulos (2018) incluem ainda considerações sobre o uso de métodos combinados nos quais diferentes métodos são utilizados sobre a mesma série temporal e uma média dos resultados é calculada. De acordo com Clemen (1989 apud HYNDMAN; ATHANASOPOULOS, 2018), combinar múltiplos *forecasts* aumenta a acurácia da predição e, em muitos casos, usar uma média simples dos valores combinados de diferentes métodos já é o suficiente para se alcançar uma predição melhor. Gooijer e Hyndman (2006) comentam que a média simples é o método de combinação mais usado e, além desse, apresentam considerações sobre o uso de uma mistura linear de *forecasts* individuais com pesos combinados e comentam sobre o uso de um sistema *fuzzy* para combinar um conjunto de *forecasts* individuais de forma não-linear.

Os métodos estatísticos, entre os quais, ARIMA, Theta (ASSIMAKOPOULOS; NIKOLOPOULOS, 2000) e outros exemplos, geralmente apresentam excelente desempenho no processo de predição. Isso pode ser observado em competições internacionais como a competição M3 (MAKRIDAKIS; HIBON, 2000) em que as equipes competidoras tiveram que propor métodos para a predição de 3.003 séries de diferentes domínios. Entre os concorrentes, os métodos podem ser divididos nas categorias "*Naïve/simple*", "*Explicit trend models*", "*Decomposition*", "*ARIMA/ARARMA model*", "*Expert System*" e "*Neural networks*". Os resultados mostram que métodos classificados como simples, como os métodos Single (BROWN, 1963) ou o método de Suavização Exponencial de Tendência Amortecida

(*Damped Trend Exponential Smoothing*) (MCKENZIE; GARDNER, 2010), são capazes de apresentar resultados competitivos com métodos mais sofisticados como os modelos ARIMA e ARARMA (PARZEN, 1982). O melhor resultado geral foi apresentado por uma equipe que utilizou o método Theta, um método da categoria Decomposição. Outro método utilizado e que demonstrou bons resultados foi o ForecastPro (GOODRICH, 2000), da categoria *Expert System*, que faz a seleção de diferentes métodos: Suavização Exponencial, Box-Jenkins, modelos binomiais negativos e Poisson, método de Croston e média móvel simples.

Em relação aos dados dessa mesma competição M3, Ahmed et al. (2010) faz uma análise de um subconjunto de 1045 séries mensais utilizando para a predição métodos de inteligência artificial/aprendizagem de máquina, a saber: *Multilayer Perceptron* (MLP), *Bayesian Neural Network* (BNN), *Radial Basis Function Neural Network* (RBF), *Generalized Regression Neural Network* (GRNN), *K-Nearest Neighbor Regression* (KNN), *Classification & Regression Trees* (CART), *Support Vector Regression* (SVR), *Gaussian Processes* (GP). Isso demonstra a diversidade de métodos que podem ser utilizados na predição de séries temporais e indica que, certamente, muitos outros modelos surgirão com o tempo e com a necessidade cada vez maior de predições mais precisas e calculadas, cada vez mais, para um conjunto maior de dados.

Numa edição mais recente da mesma competição, então denominada M4, (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a), com um conjunto de 100.000 séries temporais, nota-se por sua vez, um incremento no uso de métodos combinados. Dos 17 métodos com melhor desempenho, 12 fazem uso de combinações. E a grande surpresa, conforme destacado pelos autores em (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a), foi um método híbrido (métodos estatísticos e de *machine learning*), denominado de *Exponential Smoothing - Recurrent Neural Network* (ES-RNN) (SMYL, 2018), que obteve a melhor colocação da competição.

Além dos modelos apresentados anteriormente, os autores em (ADHIKARI; AGRAWAL, 2013) citam outros métodos de predição que aparecem de forma expressiva na literatura, que incluem:

- Modelos baseados em **Redes Neurais Artificiais** (ANN, Artificial Neural Network)
- Modelos baseados em **Máquinas de Vetores de Suporte** (SVM, Support Vector Machine)

Isso demonstra haver um grande potencial para a proposição de novos métodos, para encontrar técnicas que venham cada vez mais conseguir identificar as características de uma série temporal, suprimir ruídos e interferências existentes e estimar com maior precisão eventos futuros da série.

2.2 Multisséries Temporais

Uma série temporal pode ser descrita como uma sequência de eventos distribuídos uniformemente no tempo; ou seja, em que os eventos da série são observados numa frequência constante. Nessa definição, não é levado em consideração se tais eventos são influenciados por fatores externos que não façam parte dos dados originais da série.

Porém, em muitas situações reais, é comum termos a percepção de que fatores externos, relativos a eventos que não fazem parte das observações naturais da série, influenciam, de certa forma, o comportamento das ocorrências medidas para determinada série. Se observarmos, por exemplo, um volume anormal de chuvas numa determinada região, é provável que as plantações locais sejam afetadas por isso e que o mercado comprador desses produtos perceba alteração nos preços de verduras, legumes, frutas e cereais, em decorrência desse evento meteorológico. Outro exemplo, agora em relação à internet, poderia ser o fato de se observar uma busca maior sobre determinado tema quando ele está em destaque na mídia. Isso pode exigir um consumo maior de energia e de recursos em um conjunto de servidores que antes não eram acessados nessa mesma escala.

A análise conjunta de séries temporais pode ser de interesse para verificar se existem características ou atributos comuns entre as séries de forma a agrupá-las por algum grau de similaridade, ou pelo interesse na padronização do método a ser adotado para sua análise, a fim de ter ganho no processo de predição de diferentes séries de diferentes eventos de forma agrupada. Tanto para os casos de multisséries com alguma relação entre seus eventos, ou para multisséries independentes entre si, o interessante é identificar métodos de extrair suas características e correlações de forma a obter predições de seus eventos de maneira mais eficiente procurando evitar, sempre que possível, a necessidade de uma análise individual (personalizada) de cada série.

Outro ponto a destacar é que cada algoritmo de série temporal possui características próprias, tornando-os mais aptos para algumas séries do que outras. Dessa forma, cenários de multisséries temporais também são úteis para se avaliar a generalidade dos algoritmos, devido à diversidade das características das séries. Para isso, alguns conceitos são de relevância para o seu estudo e são descritos brevemente a seguir.

2.2.1 Técnicas de Pré-processamento

Para o processo de treinamento e predição, muitos métodos podem fazer a análise dos dados em sua forma original, sem necessidade de adequações ou modificações no estado geral das séries temporais ou no intervalo de valores dos dados. No entanto, para algumas abordagens, para determinados algoritmos, o desempenho pode ser beneficiado pela aplicação de algumas alterações nos dados.

Essas modificações podem ser implementadas com o uso de diferentes métodos,

sendo os mais comuns para o processamento de séries temporais:

- Normalização e Padronização:
  - Num problema com múltiplas séries temporais, as escalas de valores das observações das séries podem variar enormemente entre elas. Isso ocorre, porque enquanto uma série pode conter dados da variação de temperatura de uma determinada cidade, outra série pode conter valores referentes a transações monetárias na faixa de milhões.
  - O uso de técnicas como as de normalização e de padronização evita que os algoritmos precisem ajustar seu modelo de treinamento para diferentes faixas de valores.
  - Com a normalização (*normalization*) os valores de uma série são ajustados para uma faixa de valores que varia de 0 a 1.
  - A padronização (*standardization*) dos valores, por sua vez, ajusta os valores das séries para uma faixa padronizada fazendo uso da média de valores da série e do desvio padrão apresentado pelos dados (TROTTA, 2018).
- Dessazonalização (*deseasonalization*):
  - A dessazonalização corresponde a uma técnica que busca eliminar os efeitos da sazonalidade nas séries. Ela pode ser obtida, por exemplo, pelo método da decomposição clássica assumindo um relacionamento multiplicativo da componente sazonal (FIORUCCI, 2016).
- Diferenciação:
  - Aplicar a diferenciação numa série corresponde a criar uma nova série cujos dados são obtidos a partir do cálculo da diferença entre cada elemento da série e o seu antecessor.
  - É um modo de remover tendências e tornar a série estacionária (BROWNLEE, 2017).
- Alteração/Ajuste de frequência:
  - O ajuste de frequência pode ser aplicado quando as observações da série estão registradas de forma irregular, geralmente pela falta de dados na série, e se observa a necessidade de compensar os dados.
  - Ou então, a frequência de uma série pode ser alterada para que se possa, por exemplo, aplicar sobre essa série um modelo de aprendizagem que tenha sido treinado com séries de outra frequência.

  - O método de predição proposto por (PAWLIKOWSKI; CHOROWSKA, 2020) é um exemplo de abordagem que faz uso dessa técnica (denominada de *frequency change*).

- Corte (*trimming*):

  - O corte nas séries pode ser aplicado, por exemplo, quando o algoritmo de predição em uso depende que as séries tenham o mesmo comprimento, ou com o objetivo de eliminar dados da série, geralmente dados iniciais (mais antigos) da série.
  - O *trimming* pode ser justificado, também, para eliminar valores zerados das séries, como valores faltantes, ou muitas vezes gerados pela transformação de dados e que, geralmente, estão no começo ou no final da série.
  - Os autores em (PAWLIKOWSKI; CHOROWSKA, 2020) fazem uso dessa técnica em seu método de predição.

- Transformação Box-Cox:

  - A transformação Box-Cox (BOX; COX, 1964) ajusta a distribuição dos dados de uma série para uma distribuição normal e, com isso, é capaz também de fazer ajustes na variância dos dados de uma série temporal.

Outras atividades na preparação dos dados das séries podem incluir o preenchimento de valores faltantes das séries, a eliminação de *outliers*, o aumento na quantidade dos dados da série, a adição de variáveis fictícias (*dummy variables*) e a criação de atributos que auxiliem no processamento da série temporal em análise.

A autora em (TROTTA, 2018), por exemplo, cita que faz uso de dados sintéticos inserindo dados no início das séries com o objetivo de padronizar o tamanho das séries analisadas. O uso de *dummy variables* em (BONTEMPI, 2018) e (DOORNIK; HENDRIK; CASTLE, 2018) auxiliam no tratamento da sazonalidade. Ou seja, a técnica de pré-processamento a ser adotada depende do método selecionado e das condições do conjunto de séries a serem analisadas.

2.2.2 Modelos de Aprendizagem

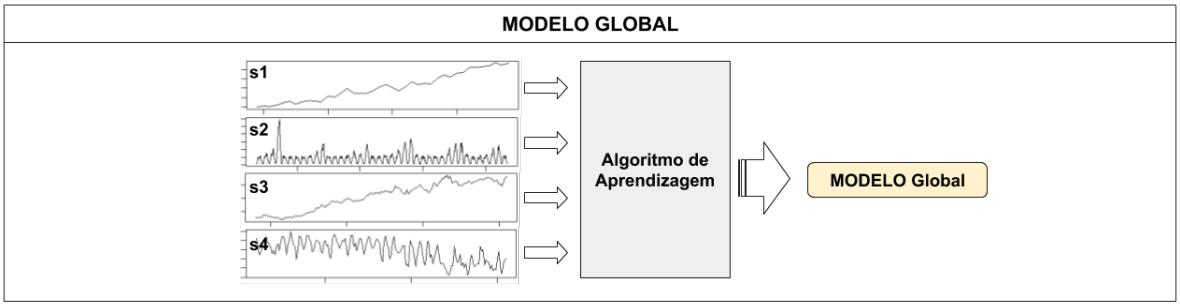
Um algoritmo de aprendizagem de máquina ao fazer a análise de um conjunto de dados, seja para uma aprendizagem supervisionada ou não-supervisionada, estabelece um modelo de representação dos dados que auxiliará posteriormente no processo de predição ou classificação de novos dados.

Na análise de um único conjunto de dados (por exemplo: uma série temporal), geralmente é estabelecido um modelo de aprendizagem para cada algoritmo a que o *dataset*

seja submetido. Mas, para problemas de multisséries temporais, diferentes abordagens podem ser consideradas, como os modelos global, local, para um grupo de séries, ou o modelo de meta-aprendizagem (*metalearning*).

No **modelo global** (Figura 3), o processo de treinamento aplicado sobre as múltiplas séries gera um único modelo que, depois, será utilizado para o processo de predição de todas as séries. Nessa abordagem, um único modelo deve ser capaz de incorporar de forma concentrada diferentes características de todas as séries que foram utilizadas durante o treinamento. A vantagem de um modelo como esse é que, em geral, o processo de aprendizagem tem a possibilidade de absorver as diferentes características das séries que são usadas no treinamento, tornando o modelo robusto, ou mais experiente para o processo de predição. Os autores em (SMYL; RANGANATHAN; PASQUA, 2018) propõem um método híbrido com os algoritmos de suavização exponencial e redes neurais recorrentes (Recurrent Neural Network, RNN) e comentam sobre o uso de um modelo hierárquico global para as redes neurais e a definição de parâmetros por série relativos à sazonalidade e suavização exponencial. No método proposto por (FILHO, 2018), múltiplas séries temporais são agrupadas num único arquivo (*melted*) com a justificativa de que os algoritmos de aprendizagem de máquina podem apresentar melhores resultados quando utilizam maior volume de dados no processo de treinamento.

Figura 3 – Modelo de Aprendizagem Global. Modelo de aprendizagem único desenvolvido a partir da análise de todas as séries do dataset.

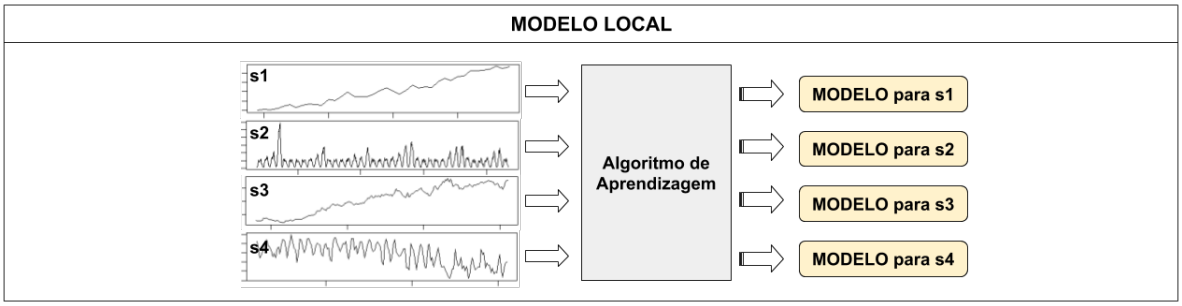


Fonte: Autoria própria.

No **modelo local** (Figura 4) o treinamento realizado gera um modelo individual para cada série do *dataset*. Isso corresponde a dizer que, para cada série, é gerado um modelo customizado, capaz de compreender em maior profundidade as características individuais da série. Tem como desvantagem, além da criação de múltiplos modelos, que o processo de treinamento tem que ser estabelecido para todas as séries, sem possibilidade de reuso para novas séries. Os autores em (PAWLIKOWSKI; CHOROWSKA; YANCHUK, 2018) utilizam o processamento individual de cada série em seu método de predição que submete a série a diferentes algoritmos a fim de estabelecer o peso que cada um

terá na definição da predição final calculada para a série. Ou seja, para cada série um estudo personalizado de suas características é efetuado de modo a selecionar os melhores parâmetros para o seu processamento.

Figura 4 – Modelo de Aprendizagem Local. Modelo de aprendizagem individual gerado para cada série do *dataset*.



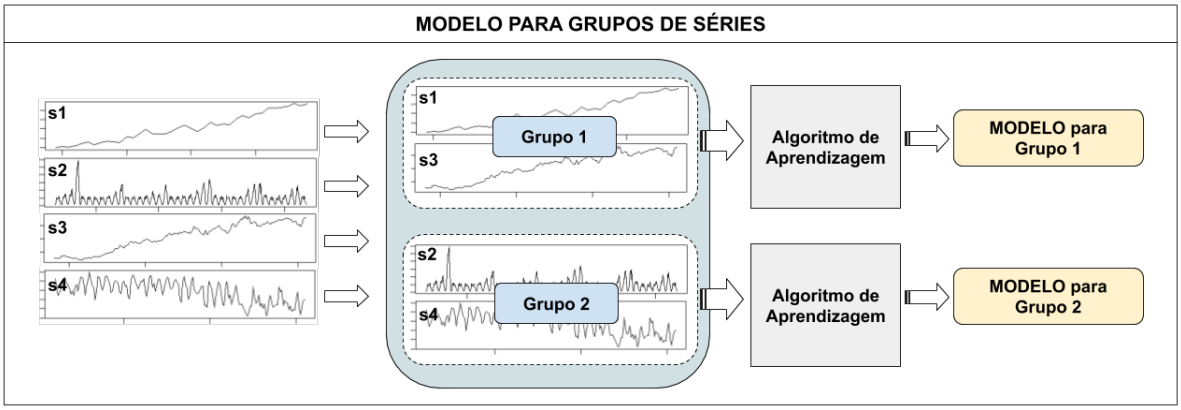
Fonte: Autoria própria.

No **modelo para um grupo de séries** (Figura 5) considera-se que as séries que compõem o conjunto de dados a ser analisado podem ser agrupadas (*clustering*) segundo diferentes critérios e, nesse caso, os modelos de aprendizagem podem ser estabelecidos para cada um dos grupos criados, uma vez que as séries de cada *cluster* têm atributos em comum que estabelecem alguma semelhança entre elas. Esses agrupamentos das séries podem ser feitos sob diferentes critérios, seja com base em características extraídas das séries individualmente, como a sua frequência, o tamanho, a entropia, a linearidade, a correlação entre os elementos das séries, entre outras características, ou por algum atributo externo da série como a categoria de dados que contêm (por exemplo: dados demográficos, dados econômicos, dados climáticos, etc.).

Como os modelos são estabelecidos por grupos de séries, pode-se denominar tais modelos de meta-modelos, uma vez que são construídos a partir de atributos dos grupos aos quais às séries estão associadas. O método proposto por (TROTTA, 2018) faz uso de uma abordagem em que os modelos são criados para grupos diferentes de séries, separados pela frequência (anual, mensal, diária) das séries. No caso, criam-se três modelos para cada grupo de forma a treiná-los com diferentes intervalos de dados, múltiplos da periodicidade da série. Para séries mensais, por exemplo, treinam-se para as periodicidades de 48, 120 e 240 meses. O autor destaca que o uso da abordagem de modelos para grupos de séries, é útil em problemas de séries temporais, especialmente porque, em geral, o volume de dados para treinamento em cada série é muito pequeno.

Num **modelo de *metalearning*** (Figura 6), os dados utilizados no treinamento de uma série, ou conjunto de séries, são usados como base para a predição de outras séries não avaliadas durante o processo de *fitting* (treinamento). No método proposto por

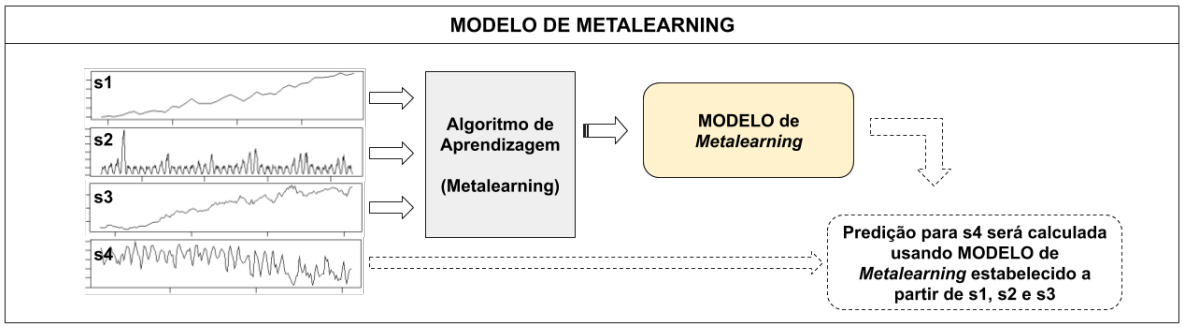
Figura 5 – Modelo de Aprendizagem para Grupos de Séries. Modelo de aprendizagem é criado para séries agrupadas por algum critério de similaridade.



Fonte: Autoria própria.

(MONTERO-MANSO et al., 2018a), por exemplo, o modelo que estabelece o peso que as predições de diferentes algoritmos terão no cálculo da predição final de uma séries é definido com base em um conjunto pequeno das séries analisadas. Ele é estabelecido com base em características extraídas das séries (metadados).

Figura 6 – Modelo de Aprendizagem de *Metalearning* (Meta-aprendizagem). O modelo de aprendizagem é criado com base em características de um grupo de série e depois é utilizado na predição de outras séries.



Fonte: Autoria própria.

2.2.3 Predição para Horizontes Mais Distantes

O processo de *forecasting* de uma série temporal tem como objetivo fazer a predição de eventos futuros para uma determinada série com base na aprendizagem obtida durante o processo de treinamento.

Para um problema de predição, normalmente se define o horizonte de predições a serem entregues pelo processo. Para séries mensais, por exemplo, isso corresponde a indicar quantos meses devem ser preditos além dos elementos originais da série. O horizonte de predição pode ser, por exemplo, de 1 mês, de 12 meses, ou de qualquer outra quantidade esperada pelo problema que está sendo resolvido.

De acordo com os autores em (TAIEB; SORJAMAA; BONTEMPI, 2010), a predição para diversos horizontes pode ser obtida de forma **iterativa**, um mês de cada vez (*one-step-ahead forecasting*), por exemplo, ou então de forma **direta**, na qual se obtêm os valores de predição para múltiplos meses numa única vez, denominada também de *multi-step-ahead forecasting*. Em relação à abordagem iterativa, em que *forecasts* são realizados mês a mês, os autores destacam que ela pode apresentar baixa performance, especialmente para horizontes maiores. Além disso, ela está sujeita a erros acumulados uma vez que os dados de uma predição (ao invés de dados originais da série) são utilizados na predição para o próximo horizonte. Em relação ao método direto, por sua vez, destacam que ele é menos propenso à acumulação de erros de predição.

Diferentes tecnologias de predição podem usar diferentes abordagens. No caso de uso de um algoritmo como o auto.arima, por exemplo, a função *forecast::auto.arima()* faz gerações iterativas de predições (*one-step-ahead forecasting*); ou seja, realiza a predição para um mês de cada vez. Apesar disso, a função devolve todas as predições de uma única vez.

Na abordagem apresentada por (SMYL; RANGANATHAN; PASQUA, 2018), por sua vez, as predições para todos os horizontes esperados são calculadas ao mesmo tempo com o uso de um método que utiliza, entre outras técnicas, redes neurais recorrentes (RNN, *Recurrent Neural Networks*).

2.2.4 Exemplos de Métodos de Predição Aplicados a Problemas de Multissséries Temporais

Os autores em (DAIGO; TOMOHARU, 2012) utilizam algoritmos genéticos com o objetivo de identificar a correlação entre ações de diferentes companhias e prever tendências de aumento ou diminuição de preços de tais ações. Com a observação de variações nos valores de ações de múltiplas companhias e artigos em jornais eles propõem uma estratégia para analisar como tais variações afetam a predição de variações nas ações de uma determinada empresa.

Em (KHAN et al., 2012), por sua vez, os autores fazem uso de Cadeias de Markov, mais especificamente Modelos Ocultos de Markov (HMM, *Hidden Markov Models*), para identificar correlações em *clusters* de máquinas virtuais e prever variações na carga de servidores na nuvem (*cloud*).

O autor em (LEI, 2013) propõe o uso de Análise de Espectro Singular (SSA, *Singular Spectrum Analysis*) que, de acordo com o autor, engloba elementos de análise de séries temporais, estatística e geometria multivariada, sistemas dinâmicos e processamento de sinal. A técnica tem por objetivo fazer a predição de variação cambial de Libra/Dólar e toma como referência a variação ajustada das moedas Euro e Dólar.

Já em (WALGAMPAYA; KANTARDZIC, 2006), os autores fazem um estudo que busca otimizar o número de sensores a serem instalados numa região para a medição/estimativa de energia consumida. Para isso, fazem uma análise comparativa de predição de séries temporais com três diferentes técnicas a fim de selecionar o melhor método: Máquinas de Vetor de Suporte (SVM, *Support Vector Machines*), *Multilayer Perceptron* (MLP) e Regressão Múltipla (MR, *Multiple Regression*).

Com base nesses exemplos e no que já foi exposto nesta seção sobre Séries Temporais, pode-se observar que são inúmeras as técnicas que podem ser utilizadas na ánálise de problemas dessa natureza.

A fim de apresentar outros exemplos de abordagens, são apresentadas a seguir algumas técnicas utilizadas nas competições M3 (MAKRIDAKIS; HIBON, 2000) e M4 (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a), que se referem a competições de predição de multisséries temporais univariadas com um conjunto de 3.003 e de 100.000 séries, respectivamente. Esses conjuntos de séries estão sendo mencionados nesta seção por serem objetos de análise neste estudo e serão detalhados posteriormente.

2.2.5 Exemplos de Técnicas de Predição Destacadas pelas Competições M3 e M4 para o *Forecasting* de Multisséries Temporais

O objetivo geral deste estudo é o de apresentar uma proposta de método de predição de séries temporais univariadas que utilize algoritmo de mineração de fluxos de dados. Para os experimentos de apoio ao desenvolvimento do método, são utilizadas bases de dados de séries temporais das competições M3 e M4. Com base nisso, os resultados apresentados pelas equipes participantes desses eventos são os motivadores pela busca de um método que apresente performance competitiva.

Nesta seção, são apresentadas breves descrições sobre alguns dos métodos que tiveram maior destaque nas competições M3 e M4, com o objetivo de servirem de exemplos de diferentes abordagens que podem ser utilizadas na predição de multisséries temporais.

Competição M3

A competição M3 apresenta os resultados conforme diferentes indicadores. Então, para este estudo, é apresentada uma descrição do método que mais se destacou entre os

demais quando analisado sob diferentes tipos (categorias) de séries temporais e diferentes frequências.

MÉTODO Theta:

De acordo com (HYNDMAN; BILLAH, 2003), o método Theta (ASSIMAKOPOULOS; NIKOLOPOULOS, 2000) pode ser comparado a um método de suavização exponencial simples com *drift* (*Simple Exponential Smoothing with drift, SES-d*) [2] e faz uso de técnicas de decomposição, seguidas de projeção e combinação de componentes individuais.

Em (FIORUCCI, 2016), o autor enumera os passos do método Theta da seguinte forma:

1. Dessazonalização (*Deseasonalisation*): consiste na eliminação dos efeitos da sazonalidade. Para isso, a sazonalidade é eliminada usando o método de decomposição multiplicativa clássica.

2. Decomposição: a série é decomposta em duas linhas theta (a linha de regressão linear *Z(0)* e a linha theta *Z(2)*).

3. Extrapolação: *Z(0)* é extrapolada como uma linha de regressão linear normal e *Z(2)* é extrapolada usando SES (*Simple Exponential Smoothing*).

4. Combinação: a predição final é a combinação dos *forecasts* de duas linhas *theta* usando pesos iguais.

5. Devolução da sazonalidade à série (*Reseasonalisation*): se a série apresentava sazonalidade no passo 1, então o *forecast* final gerado pelo método é multiplicado pelos índices de sazonalidade determinados para a série.

Os autores em (HYNDMAN; BILLAH, 2003) fazem um detalhamento didático do método Theta, apresentando-o com uma notação matemática mais simples que a original de (ASSIMAKOPOULOS; NIKOLOPOULOS, 2000). Nessa referência os autores apresentam as seguintes explicações e fórmulas:

[2] No método SES-d, o termo *drift* corresponde a uma correção aplicada no modelo de suavização que define o grau de inclinação (ou crescimento, *growth*) de uma regressão linear simples da série. Na equivalência entre Theta e SES-d, esse *drift* é estabelecido como metade do coeficiente de inclinação da regressão (FIORUCCI et al., 2016) (NIKOLOPOULOS et al., 2011). O termo *drift* aplicado ao método SES-d está explicado neste trecho para desambiguação em relação ao termo *concept drift* (ou *drift*) que será apresentado na seção de Mineração de Fluxos de Dados deste estudo.

Para uma série temporal univariada $X_1, ...X_n$, é construída uma série $Y_{1,\theta}, ...Y_{n,\theta}$ para $t = 3, ..., n$. (Equação 2.9)

$$Y''_{t,\theta} = \theta X''_t \tag{2.9}$$

onde X''_t corresponde à segunda diferenciação de X_t e $Y''_{t,\theta}$ corresponde à segunda diferenciação de $Y_{t,\theta}$. Essa equação, segundo (HYNDMAN; BILLAH, 2003) é uma equação diferencial de segunda-ordem e pode ser expressa como (Equação 2.10):

$$Y_{t,\theta} = a_\theta + b_\theta(t - 1) + \theta X_t \tag{2.10}$$

onde a_θ e b_θ são constantes e $t = 1, \ldots, n$. Na Equação 2.10 $Y_{t,\theta}$ é equivalente a uma função linear de X_t com a adição de uma tendência linear. Os autores em (HYNDMAN; BILLAH, 2003) dizem que, de acordo com (ASSIMAKOPOULOS; NIKOLOPOULOS, 2000), $Y_{t,\theta}$ é uma "linha Theta".

No método Theta, as predições são obtidas por uma média ponderada de *forecasts* de $Y_{t,\theta}$ para diferentes valores de θ. A predição para um horizonte h baseado nas observações $X_1, ..., X_n$ é dada pela Equação 2.11:

$$\hat{X}_n(h) = \frac{1}{2}[\hat{Y}_{n,0}(h) + \hat{Y}_{n,2}(h)] \tag{2.11}$$

onde $\hat{Y}_{n,0}(h)$ é obtido extrapolando a parte linear da Equação 2.10 e $\hat{Y}_{n,2}(h)$ é obtido usando a suavização exponencial simples na série $Y_{t,2}$, conforme descrito a seguir (Equações 2.12 e 2.13):

$$\hat{Y}_{n,0}(h) = \hat{a}_0 + \hat{b}_0(n + h - 1) \tag{2.12}$$

e

$$\hat{Y}_{n,2}(h) = \alpha \sum_{i=0}^{n-1} (1 - \alpha)^i Y_{n-i,2} + (1 - \alpha)^n Y_{1,2} \tag{2.13}$$

onde α é o parâmetro de suavização para o algoritmo SES (Suavização exponencial simples).

Competição M4

Em relação à competição M4, destacam-se os métodos que foram os 3 melhores colocados na classificação geral, que incluem: 1 método híbrido e dois métodos que fazem uso da estratégia de combinação.

MÉTODO *Exponential Smoothing - Recurrent Neural Network (ES-RNN) Hybrid Models*:

De acordo com os autores do método (SMYL, 2018) e (SMYL; RANGANATHAN; PASQUA, 2018), o ES-RNN faz uso de um modelo híbrido em que os coeficientes de sazonalidade e suavização de um algoritmo de Suavização Exponencial (*Exponential Smoothing*) são treinados de forma concorrente com os pesos de uma Rede Neural usando o mesmo método de Descida de Gradiente para ambos. Os modelos de treinamento são considerados hierárquicos por definirem, de forma global, os pesos para as redes neurais e, de forma local (para cada série, de forma individual), os coeficientes da suavização exponencial.

Conforme os autores, o uso do método de suavização exponencial é justificado por ser uma técnica reconhecida para lidar com a sazonalidade como parte do modelo de treinamento.

Para o pré-processamento das séries, o método faz uso de janelas deslizantes definindo, geralmente, uma janela de entrada no comprimento da sazonalidade da série (por exemplo: 12 para séries mensais) e uma janela de saída com o tamanho do horizonte de predição esperado (no caso, 18 meses para séries mensais). Os autores sugerem que a normalização dos dados é importante, porém não fazem a normalização da série como um todo, mas sim com o uso das janelas deslizantes, normalizando os dados com um coeficiente (por exemplo: a média dos itens da janela de entrada).

Quanto à eliminação da sazonalidade (*deseasonalization*), os autores do método preferem a abordagem em que ela é feita junto com o processo de *forecasting*, tal como é feito pelos algoritmos de suavização exponencial. Para os modelos sazonais, os autores usam algoritmos baseados na sazonalidade multiplicativa de Holt-Winters e apresentam as Equações 2.14 e 2.15:

$$l_t = \alpha * y_t / s_t + (1 - \alpha) * l_{t-1} \tag{2.14}$$

e

$$s_{t+K} = \beta * y_t / l_t + (1 - \beta) * s_t \tag{2.15}$$

onde l_t corresponde ao nível, y_t contém os valores das séries e s_t é o coeficiente de sazonalidade no tempo t. K é o tamanho da sazonalidade (exemplo: 18 para séries mensais), α e β são os coeficientes de suavização (entre 0 e 1) e $s_t > 0$.

Os autores definem como equação para o *forecasting* (Equação 2.16):

$$\hat{y}_{t+1 \ldots t+h} = RNN(x_t) * l_t * s_{t+1 \ldots t+h} \tag{2.16}$$

onde x_t é o vetor de entrada pré-processado (normalizado e sem sazonalidade). As predições feitas por RNN são feitas ao mesmo tempo. No caso de séries mensais, as 18 predições são entregues de uma única vez.

Uma curiosidade é que o método faz uso apenas dos dados dos 20 anos mais recentes de cada série. Ou seja, para séries longas (considerando que a competição tem séries mensais com até 2.794 observações), apenas parte dos dados das séries é considerada.

MÉTODO M4metalearning:

O método M4metalearning utiliza uma abordagem de combinação descrita por seus autores em (MONTERO-MANSO et al., 2018a) e (MONTERO-MANSO et al., 2018b).

As predições geradas pelo método são o resultado de uma combinação das predições de diferentes métodos estatísticos e um método de *machine learning*, em que os pesos atribuídos para cada algoritmo são calculados para cada série usando um modelo de aprendizagem baseado em *Gradient Tree Boosting* (no caso, *xgboost* (CHEN et al., 2018)).

Os algoritmos de predição usados pelo método são os seguintes:

- ARIMA (auto.arima)
- ETS (*Exponential Smoothing State Space model*)
- NNETAR (uma rede neural *feed-forward*)
- TBATS (*Exponential Smoothing State Space Trigonometric, Box-Cox transformation, ARMA errors, Trend and Seasonal components model*)
- STLM-AR (*Seasonal and Trend decomposition using Loess with AR modeling of the seasonally adjusted series*)
- RW-DRIFT (*Random Walk with Drift*)
- THETAF
- NAIVE
- SEASONAL NAIVE (modelo em que o *forecast* considera o último valor observado do mesmo período).

O método faz uso de um modelo de *metalearning* que define qual o peso que cada algoritmo terá na predição. Esse modelo é criado a partir de um subconjunto das séries e testado com séries não usadas anteriormente para evitar *overfitting* e faz uso de 42 *features* extraídas das séries para a definição do metamodelo. As características identificadas nas séries incluem informações como coeficientes de autocorrelação, entropia, linearidade,

sazonalidade, curvatura, estabilidade, comprimento das séries, entre outros (uma lista com a descrição das 42 características pode ser encontrada no Apêndice A, Experimento 2.9 deste estudo, no link http://bit.ly/37yBGw2).

Uma vez que o modelo de *metalearning* já tenha sido criado, podem ser realizadas as predições das séries. Para isso, o vetor com as 42 *features* é obtido para a série em análise e são realizadas predições para a série com os algoritmos do método. O *forecast* resultado da combinação linear das predições individuais dos métodos é calculado conforme a Equação 2.17 apresentada pelos autores do método (observar que, para simplificação, a fórmula disponibilizada pelos autores apresenta a forma de predição para o horizonte de 1 mês, para o caso de séries mensais):

$$combi\_forecast(x) = \sum_{i}^{M} \frac{e^{y(a)_i}}{e^{y(a)}} f_i \tag{2.17}$$

Ou seja, para uma série x, um vetor de *features* (α) é criado e, em seguida, são processados os *forecasts* individuais (f) para a série usando os algoritmos estatísticos e de *machine learning*. O método produz uma saída denominada $y(a)$ que é transformada em probabilidades e os *forecasts* (de i a M) são combinados conforme a equação apresentada.

Por suas características de *metalearning*, esse método é utilizado em alguns experimentos desenvolvidos ao longo deste estudo (ver Apêndice A).

MÉTODO *Weighted Ensemble of Statistical Models*:

O método *Weighted Ensemble of Statistical Models* utiliza uma abordagem de combinação descrita por seus autores em (PAWLIKOWSKI; CHOROWSKA; YANCHUK, 2018) e (PAWLIKOWSKI; CHOROWSKA, 2020), que inclui cinco passos, executados para cada série:

1. Escolher um conjunto de algoritmos a combinar. A escolha leva em conta a periodicidade (frequência) e a existência ou não de sazonalidade e tendência.

2. Usando os algoritmos selecionados, criar *forecasts* para um horizonte de predição (*one-step-ahead forecasts*) para os últimos N registros da série e medir a acurácia das predições usando o sMAPE (ARMSTRONG, 1985). O valor de N, de acordo com os autores, varia conforme a frequência da série, mas geralmente é inferior ao total de *forecasts* a serem calculados para a série (no caso de séries mensais, o número de predições esperadas é igual a 18 meses).

3. Para cada algoritmo, fazer uma análise dos erros obtidos com o objetivo de se ter um histórico dos erros produzidos pelo algoritmo para cada grupo de séries e determinar um "*score*".

4. De acordo com os resultados apresentados pelos algoritmos (no passo 3), determinar os pesos de cada algoritmo. O peso depende da periodicidade do grupo e geralmente é calculado como o quadrado do inverso do *score* obtido em 3.

5. Calcular o *forecast* final para a série, que corresponde a uma média ponderada dos *forecasts* usando os pesos determinados no passo 4.

Em um documento posterior à competição M4 (PAWLIKOWSKI; CHOROWSKA, 2020), os autores ajustam a descrição do método para:

1. Classificar a série temporal em *clusters* (as séries são agrupadas com base na frequência e na e na existência ou não de tendência e sazonalidade).

2. Escolher o *pool* de algoritmos para cada *cluster*.

3. Medir a performance dos modelos usando a técnica de *Rolling Origin Evaluation* (TASHMAN, 2000).

4. Determinar os pesos de cada algoritmo no *pool*.

5. Calcular o *forecast* final.

As séries são classificadas com base na existência de tendência e/ou sazonalidade. Cada série é associada a um diferente conjunto de algoritmos de predição. O método inclui a execução de experimentos para identificar quais os algoritmos melhor aplicados a cada tipo de série e também para definir os pesos de cada modelo.

Os algoritmos de predição usados pelo método são listados a seguir com a indicação, quando disponível, da implementação em R utilizada (usando a notação "*<nome do pacote R>::<nome da função>*"):

- *Naive* (Método de passeio aleatório, *random walk*, que assume que os valores futuros serão os mesmos que o da última observação conhecida)

- Naive2 (MAKRIDAKIS; WHEELWRIGHT; HYNDMAN, 1998) (método similar ao Naive, mas cujos dados são ajustados conforme a sazonalidade, pela aplicação de uma decomposição multiplicativa clássica)

- *Simple Exponential Smoothing* (Função *forecast::ses*, pacote *forecast* (HYNDMAN et al., 2019b))

- *Exponential Smoothing* (Função *forecast::ets* (HYNDMAN et al., 2019a), pacote *forecast* (HYNDMAN et al., 2019b))

- *Damped Exponential Smoothing* (*forecast::ets* (HYNDMAN et al., 2019a), *damped* = *T*, pacote *forecast* (HYNDMAN et al., 2019b))
- ARIMA (Função *forecast::auto.arima*, pacote *forecast* (HYNDMAN et al., 2019b))
- *Simple Theta* (Função *forecast::thetaf*, , pacote *forecast* (HYNDMAN et al., 2019b))
- *Optimised Theta* (Função *forecTheta::otm*, pacote *forecTheta* (FIORUCCI; LOUZADA; YIQI, 2016))
- *Econometric models* (Modelos econométricos, com uso de regressão linear para diferentes tipos de tendência)

Numa fase de pré-processamento as séries passam por eliminação de sazonalidade (*deseasonalization*), ajuste de frequência e ajuste no tamanho das séries (*trimming*).

O processamento é realizado por série, mas os parâmetros podem ser definidos de acordo com características das séries. Para séries mensais, por exemplo, o conjunto de dados é dividido em 4 grupos:

- Séries com tendência e sazonalidade.
- Séries com tendência, mas sem sazonalidade.
- Séries sem tendência, mas com sazonalidade.
- Séries sem tendência e sem sazonalidade.

Essas características do grupo vão indicar quais os métodos (algoritmos de predição) mais indicados para a série, de acordo com a existência ou não de tendência ou sazonalidade.

O *forecast* final, conforme já mencionado, é obtido com uma média ponderada das predições, onde cada algoritmo tem um peso diferente de acordo com o score atribuído a ele no momento do processo de análise.

2.3 Mineração de Fluxos de Dados

Com um volume crescente de dados gerados pelas redes sociais e pela difusão de conceitos como *IoT* e *Big Data*, o uso de técnicas tradicionais baseadas no processamento de informações em lote pode não ser eficiente para todas as áreas de aplicação. É nesse cenário que técnicas e ferramentas de Mineração de Fluxos de Dados, ou *Data Stream Mining* (DSM), estão sendo desenvolvidas.

Para a análise de séries temporais, principal interesse deste estudo, é essencial que os algoritmos utilizados sejam capazes de identificar variações de comportamento dos dados

para que o processo de predição seja cada vez mais preciso. Para essa característica, é justificável buscar a aplicação de algoritmos de mineração de fluxos de dados, que permitem um processamento gradual e incremental das observações e são altamente adaptáveis às variações apresentadas pelos dados. Além disso, como alguns problemas de séries temporais podem envolver o uso de grandes conjuntos de dados, isso sugere que o uso de técnicas de DSM pode ser aplicável.

Nas seções a seguir são apresentados conceitos relevantes ao estudo da área de mineração de fluxos de dados, iniciando com a apresentação de referências sobre o termo mudança de conceito e, em seguida, características gerais e algoritmos de DSM são explorados, em especial os relacionados a processos de regressão.

2.3.1 Mudança de Conceito (*Concept Drift*)

Um conceito importante, destacado em (GAMA et al., 2014), é o denominado de *concept drift* (mudança de conceito), que representa uma mudança na distribuição dos dados, mais precisamente mudanças na distribuição condicional das saídas (variável *target*) para uma dada entrada (atributos de entrada). No uso de algoritmos de mineração de fluxos de dados para o processamento de séries temporais, por exemplo, uma perturbação, um ruído detectado no perfil da série, poderia caracterizar um *concept drift*.

De acordo com os autores em (BIFET et al., 2018), algoritmos de mineração de fluxos de dados precisam reagir a mudanças que ocorrem nos fluxos de dados no decorrer do tempo. Essas mudanças ocorrem devido a alterações nas propriedades estatísticas dos dados que, segundo os autores, podem ocorrer naturalmente devido a flutuações casuais. Indicam que a distribuição de probabilidade observada numa série num dado momento pode não ser a mesma em outros instantes de tempo. Então, é preciso que os algoritmos de DSM sejam capazes de perceber e saber reagir a tais alterações.

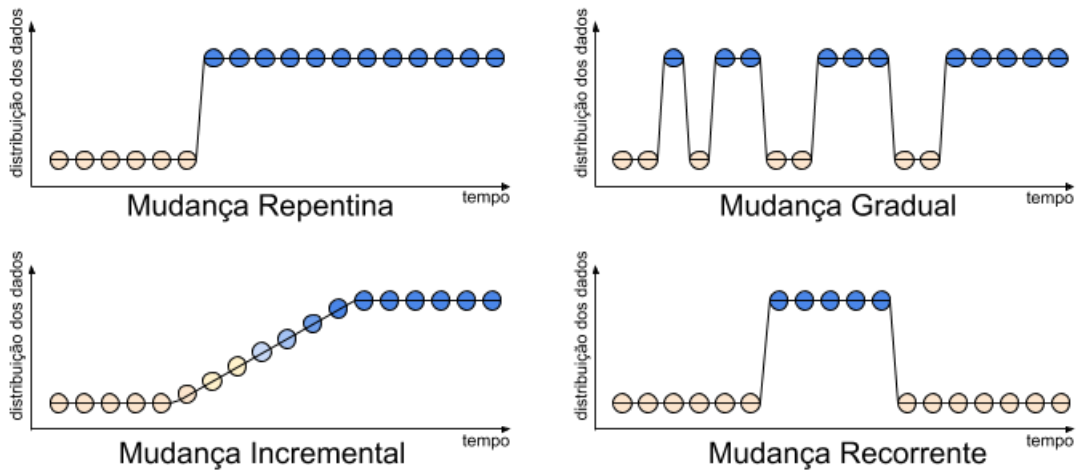
Os autores em (LU et al., 2019) fazem uma revisão de literatura sobre o tema *concept drift* com uma análise de mais de 130 publicações relacionadas ao assunto. Analisam metodologias e técnicas e propõem um *framework* para o processo de aprendizagem em cenários sob a influência de mudanças de conceito. Além disso, também apresentam e discutem 10 *datasets* de dados sintéticos bem conhecidos na literatura, assim como 14 outros *datasets* utilizados como *benchmark* para avaliar a performance de algoritmos de aprendizagem no tratamento de *concept drift*. De acordo com os autores dessa referência, a aprendizagem de máquina convencional trabalha com dois componentes principais: treinamento/aprendizagem e predição. O estudo de aprendizagem sob *concept drift* adiciona três novos componentes ao modelo de aprendizagem de máquina, a saber: detecção de mudança (ou *drift detection*, que consiste em identificar se a mudança ocorre ou não), a compreensão da mudança (ou *drift understanding*, que corresponde a identificar quando, como e onde a mudança ocorre) e a adaptação à mudança (ou *drift adaptation*, que é a

reação à mudança identificada no fluxo de dados).

Os autores em (LU et al., 2019) apresentam uma definição para *concept drift* como um fenômeno no qual as propriedades estatísticas de um domínio-alvo (*target domain*) mudam ao longo do tempo de uma forma arbitrária. Citam que a mudança de conceito ocorre em muitas situações do mundo real e exemplificam isso (de forma didática) comentando sobre a mudança na forma como utilizamos o telefone móvel ao longo do tempo. Inicialmente, os telefones celulares eram usados principalmente para chamadas de áudio e envio de mensagens de texto, mas gradualmente seu principal uso passou a ser maior como dispositivo para o acesso à internet e captura de fotos e vídeos; ou seja, houve uma mudança na distribuição da frequência do tipo de uso feito com o celular em suas diferentes finalidades.

No caso sobre o uso da telefonia celular, a mudança no comportamento ocorreu de forma gradual. Mas, às vezes, as alterações podem ocorrer de outras formas. Sobre isso, Zliobaite (2010) apresenta os seguintes tipos de mudança (Figura 7): mudança repentina, ou abrupta (*Sudden Drift*), em que uma mudança ocorre repentinamente; mudança gradual (*Gradual Drift*), em que pequenas alterações ocorrem ao longo do tempo até que a passagem para uma outra distribuição ocorra completamente; mudança incremental (*Incremental Drift*) em que a mudança vai ocorrendo gradualmente durante um período de tempo; e mudanças recorrentes (*Reoccurring Contexts*), em que uma mudança de conceito ocorre, permanece por um tempo e o estado do fluxo de dados retorna ao estado anterior após a passagem de um certo tempo.

Figura 7 – Tipos de mudança de conceito citados por Zliobaite (2010): Repentina, Gradual, Incremental, Recorrente

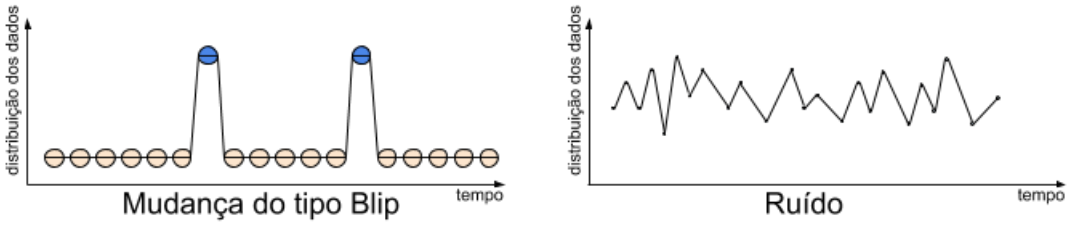


Fonte: Adaptado do original disponível em (ZLIOBAITE, 2010)

Além desses 4 tipos principais de *drifts*, é possível encontrar outras indicações na

literatura como as citadas pelos os autores em (RAMíREZ-GALLEGO et al., 2017) que incluem ainda (Figura 8): *Blip* (também conhecidos como *outliers*, que de acordo com essa referência poderiam ser ignorados visto que a mudaça ocorrida pode ser decorrente de um evento aleatório); *Noise* (Ruído), que representa a ocorrência de flutuações insignificantes no fluxo mas devem ser filtradas; e Mixed (Misto, ou misturado), em que podem ser detectados vários tipos de *drifts* no mesmo fluxo de dados em análise.

Figura 8 – Tipos de mudança de conceito citados por Ramírez-Gallego et al. (2017): Blip, Ruído



Fonte: Adaptado do original disponível em (RAMíREZ-GALLEGO et al., 2017)

Sabendo que as mudanças de conceito podem ocorrer de diferentes maneiras, é importante saber como detectá-las. Os autores em (LU et al., 2019) citam os seguintes tipos de algoritmos de deteção de *concept drift*: a detecção de *drift* baseada na taxa de erro (*Error Rate-Based Drift Detection*), a detecção baseada na distribuição dos dados (*Data Distribution-Based Drift Detection*) e a detecção de mudança baseada em teste de múltiplas hipóteses (*Multiple Hypothesis Test Drift Detection*).

Nos algoritmos baseados na análise da taxa de erro do classificador, se é identificado um aumento ou diminuição estatisticamente significativo na taxa de erro, um alarme de *drift* é disparado. Nessa categoria, os autores em (LU et al., 2019) citam que o DDM (*Drift Detection Method*) (GAMA et al., 2004) foi o primeiro algoritmo a definir o nível de alerta e o nível de mudança para detecção de um *concept drift*. Outro exemplo bem divulgado na literatura para essa categoria de algoritmo é o ADWIN (*ADaptive WINdowing*) (BIFET; GAVALDà, 2007). Por se tratarem de métodos para análise de variação de erro em fluxos de dados, as análises das taxas de erro são feitas com base nas informações contidas em janelas deslizantes de dados e cada algoritmo utiliza abordagens distintas para a seleção de dados e contabilização das taxas, de forma a comparar informações em dados recentes do fluxo de dados com dados históricos da mesma fonte de dados que está sendo processada.

Em relação aos algoritmos que fazem a detecção de *drift* baseados na análise da distribuição dos dados, os autores em (LU et al., 2019) descrevem que esses algoritmos usam uma métrica de distância para quantificar a variância (ou dissimilaridade) entre a distribuição dos dados históricos e dos novos dados que estão sendo processados. A

estratégia mais usada por esses algoritmos consiste em utilizar duas janelas deslizantes, sendo uma janela fixa que contém os dados históricos e outra móvel que contém os novos dados. Lu et al. (2019) cita que os autores em (KIFER; BEN-DAVID; GEHRKE, 2004) foram os primeiros a propor o uso de tal técnica para a detecção de mudanças em fluxos de dados.

Sobre a detecção de *drift* com o uso teste de múltiplas hipóteses, os autores em (LU et al., 2019) indicam que esse tipo utiliza técnicas similares às usadas pelas detecções baseadas em taxas de erro e distribuição dos dados. Destacam que a novidade nesses métodos é o fato de eles utilizarem testes de hipóteses múltiplas para detectar a mudança de conceito.

Antes de descrever técnicas para a detecção de mudança, os autores em (BIFET et al., 2018) apresentam critérios relevantes para avaliar o processo de detecção, a saber (observação: nessas definições θ corresponde à magnitude mínima da mudança que se quer detectar):

- *Mean time between false alarms* (MTFA): mede com que frequência é recebido um falso alarme nas situações em que nenhuma mudança é detectada. A taxa de falso alarme (*false alarm rate*, FAR), por sua vez, é definida como $1/MTFA$.
- *Mean time to detection* (MTD(θ)): mede a capacidade do sistema de aprendizagem de detectar e reagir a mudança quando ela ocorre.
- *Missed detection rate* (MDR(θ)): mede a probabilidade de não gerar um alarme nos casos em que a mudança de fato ocorre.
- *Average run length* (ARL(θ)): essa é uma medida que generaliza MTFA e MTD e que indica por quanto tempo é preciso esperar até que se detecte que uma mudança ocorreu. Nesse caso, $MTFA = ARL(0)$; e, para $\theta > 0$, $MTD(\theta) = ARL(\theta)$.

Sobre os métodos de detecção de mudança, Bifet et al. (2018) indicam que o teste de soma cumulativa (CUSUM, cumulative sum, (PAGE, 1954)) é desenhado para dar um alarme quando a média dos valores de entrada desvia-se significativamente do seu valor anterior. De acordo com (BIFET et al., 2018), CUSUM é sem memória (*memoryless*) e usa um tempo de processamento constante por item. Para uma dada sequência de observações $\{x_t\}_t$, define $z_t = (x_t - \mu)/\sigma$, onde μ é o valor esperado de x_t e σ é seu desvio padrão em condições normais; se μ e σ são desconhecidos *a priori*, eles são estimados a partir da própria sequência. Então CUSUM computa os índices e o alarme:

$g_0 = 0$,
$g_t = max(0, g_{t-1} + z_t - k)$,
Se $g_t > h$, declara mudança e reseta $g_t = 0$, μ e σ.

onde, em relação aos parâmetros k e h, a diretriz é definir k como a metade do total de mudanças que se espera detectar (medida em desvios padrões) e h como $ln(1/\delta)$, onde δ é a taxa de alarmes falsos aceitável, geralmente entre 3 e 5.

Outro método destacado pelos autores em (BIFET et al., 2018) é o teste Page-Hinkley, que é uma variação do teste de soma cumulativa CUSUM. Para o Page-Hinkley test estabelecem:

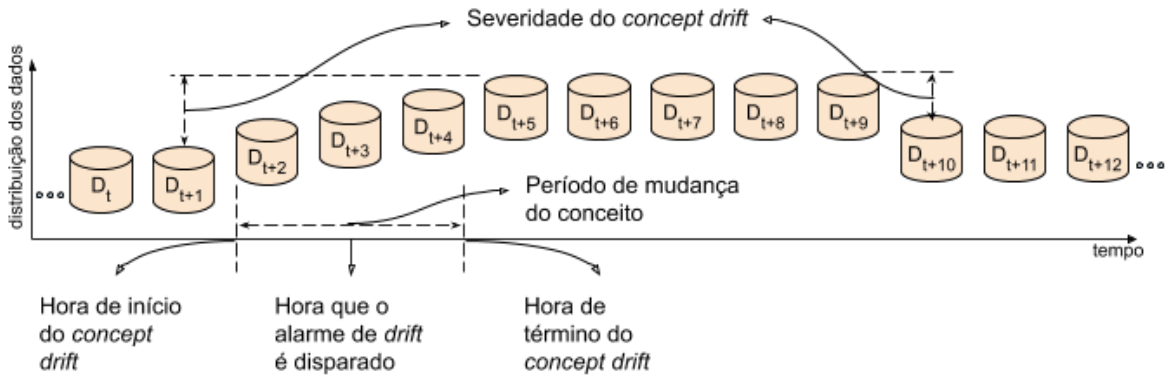
$g_0 = 0$,
$g_t = g_{t-1} + z_t - k$,
$G_t = min\{g_t, G_{t-1}\}$,
Se $g_t - G_t > h$, declara mudança e reseta $g_t = 0$, G_t, μ e σ.

Quanto aos métodos DDM e ADWIN citados anteriormente, os autores em (BIFET et al., 2018) complementam que o *Drift Detection Method* (DDM) monitora o número de erros produzidos pelo modelo de aprendizagem construído com base nos itens anteriores do fluxo. Geralmente, o erro do modelo deve decrescer ou permanecer estável à medida que mais dados são usados. Assume-se, então, que o método de aprendizagem controle o *overfitting* e que a distribuição de dados e rótulos é estacionária. Quando DDM observa que o erro de predição aumenta, isso evidencia que uma mudança ocorreu. Sobre o ADWIN, comentam que ele resolve o problema de rastrear a média de um fluxo de *bits* ou de números reais. O ADWIN mantém uma janela de comprimento variável com os itens vistos recentemente, com a propriedade de que a janela tem o comprimento máximo estatisticamente consistente com a hipótese de não ter ocorrido alteração no valor médio dos dados contidos dentro da janela. Um fragmento mais antigo da janela é descartado se e somente se houver evidências suficientes de que seu valor médio difere do resto da janela. Essa abordagem tem duas consequências: uma, de que a mudança é detectada de forma confiável mesmo que a janela encolha; e a outra, que a qualquer momento a média sobre a janela existente pode ser tomada com segurança como uma estimativa da média atual no fluxo (exceto se uma mudança muito pequena ou muito recente ainda não seja estatisticamente perceptível).

Outro aspecto importante no estudo de *concept drift* é relacionado à **compreensão da mudança** (*drift understanding*). Sobre isso, os autores em (LU et al., 2019) citam que esse componente do estudo de *concept drift* corresponde a obter informação sobre "Quando" (ou "*When*", o momento em que a mudança de conceito ocorre e por quanto tempo dura), "Como" (ou "*How*", o grau ou severidade do *drift*) e "Onde" ("Where", as regiões em que ocorre o *concept drift*). As Figuras 9 e 10 ajudam a compreender esses conceitos.

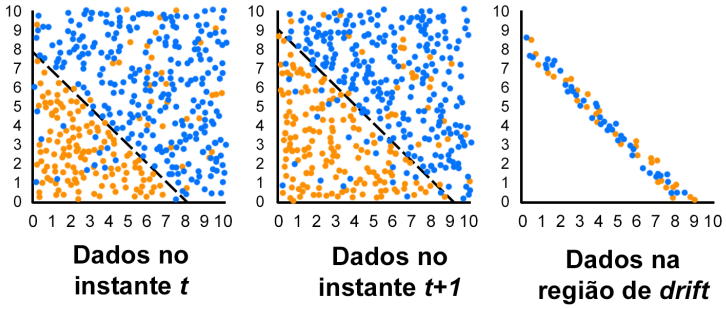
De acordo com os autores em (BIFET et al., 2018), as duas características importantes que qualquer método de classificação em mineração de fluxos de dados deve tratar

Figura 9 – Exemplo de fluxo de dados em que se identifica o momento de ocorrência e a severidade de um concept drift (LU et al., 2019).



Fonte: Adaptado do original disponível em (LU et al., 2019)

Figura 10 – Exemplo de região de ocorrência de um concept drift (LU et al., 2019).



Fonte: Adaptado do original disponível em (LU et al., 2019)

são: que existem limitações em termos de memória e tempo para o processamento (isso significa que não podemos armazenar todas as instâncias em memória) e que os modelos de classificação devem ter a capacidade de se adaptar a possíveis mudanças na distribuição dos dados que estão chegando para análise. Ainda de acordo com essa referência, um terceiro requisito é a habilidade em lidar com mudanças no espaço de atributos. A maioria dos algoritmos de classificação de fluxos de dados assume que o espaço de características é fixo e que não podem responder rapidamente a mudanças desse tipo. Além dos atributos (características) dos dados, as classes dos dados também podem mudar ao longo do tempo. Então, os algoritmos devem ser capazes de lidar com a adição de novas classes (*labels*) assim como com a deleção de classes existentes que não sejam mais representadas nos dados.

Em (BARDDAL; GOMES; ENEMBRECK, 2015) os autores abordam sobre um

tipo de *concept drift* pouco explorado na literatura que é o denominado de *Feature Drift* (mudança de característica, ou mudança de atributos), que afeta diretamente métodos baseados no processo de seleção de atributos (características). Destacam que o problema de *feature drift* pode ocorrer em diferentes cenários, mas é muito comum em casos de mineração de texto, uma vez que cada palavra num texto pode ser considerado um atributo. Então, é fácil compreender que, à medida que um fluxo de dados textuais é processado, a distribuição de probabilidade de ocorrências de diferentes palavras muda no decorrer do tempo, caracterizando uma *feature drift*.

Nesse cenário relacionado a métodos que fazem a seleção de *features*, os autores em (BARDDAL; GOMES; ENEMBRECK, 2015) citam exemplos de abordagens comumente utilizadas como Árvores de Decisão (Decision Trees) e Regras de Decisão (Decision Rules). Em relação a árvores de decisão, citam o método *Very Fast Decision Tree* (VFDT) (DOMINGOS; HULTEN, 2000), que faz a seleção de *features* para a construção de modelos no processo de aprendizagem de fluxos de dados, porém não detecta nem se adapta a possíveis mudanças de conceito ou mudanças de características. Indicam que os modelos baseados em árvores de decisão, apesar de serem de fácil compreensão, podem se tornar complexos em cenários em que as árvores tendem a crescer bastante.

Sobre os modelos baseados em Regras de Decisão, Barddal, Gomes e Enembreck (2015) indicam que esses modelos têm a vantagem de modularidade e interpretabilidade visto que cada regra pode ser interpretada isoladamente das demais. Em relação aos algoritmos de regras de decisão para classificação de fluxos de dados e que fazem seleção de *features* citam: FACIL (FERRER-TROYANO; AGUILAR-RUIZ; RIQUELME, 2005) e VFDR (*Very Fast Decision Rules*) (GAMA; KOSINA, 2011). De acordo com (BARDDAL; GOMES; ENEMBRECK, 2015) o algoritmo FACIL cria regras de forma incremental de acordo com as instâncias que vão sendo recebidas pelo algoritmo. O algoritmo implementa mecanismos implícitos e explícitos de esquecimento (remoção) de regras para o tratamento de *concept drift*. O *Very Fast Decision Rules* (VFDR), por sua vez, apresenta uma abordagem robusta para a aprendizagem de regras a partir do fluxo de dados em análise. É capaz de lidar com regras ordenadas e não ordenadas, que aumentam à medida que as instâncias são analisadas, e são criadas selecionando as *features* que minimizam a entropia da distribuição da classe em cada partição de dados.

Os autores em (BARDDAL; GOMES; ENEMBRECK, 2015) também exploram o uso de técnicas que usam aleatoriedade (*randomness*) como *Streaming Random Forest*, que é uma adaptação para o cenário de fluxos de dados do classificador *Random Forest* (BREIMAN, 2001). Outro classificador dessa categoria é o *Random Rules* que, segundo os autores, é uma adaptação do VFDR para aleatoriadade feita por Almeida, Kosina e Gama (2013). Comentam também sobre o uso de técnicas de janelamento (*windowing*) e alertam que, enquanto janelas curtas refletem a distribuição de dados atual e garantem rápida

adaptação a *drifts*, se elas forem muito curtas a performance do sistema pode ser diminuída em áreas estáveis. Em contrapartida, janelas de grande dimensão podem assegurar melhor performance em áreas estáveis dos dados, mas reação mais lenta a mudanças de conceito.

Outro tópico importante relacionado ao tema *concept drift* é o processo de **adaptação à mudança** (*drift adaptation*). Uma vez que um *drift* é detectado e avaliado, é necessário saber como reagir a essa mudança na distribuição dos dados de forma a manter o modelo de aprendizagem válido. Em algumas situações, é preciso treinar novos modelos tomando como base os dados mais recentes que estiverem disponíveis. No caso de se identificar uma mudança de conceito recorrente, Lu et al. (2019) cita que preservar e reutilizar modelos antigos pode economizar esforços que seriam necessários caso se optasse por treinar um novo modelo. Além disso, destacam que, em algumas situações, uma alternativa a um novo treinamento de um modelo completo é desenvolver um modelo que adaptativamente aprende com base nas mudanças detectadas. Ou seja, para cada situação, uma abordagem específica pode ser adotada, tendo sempre em mente que a adaptação deve ocorrer de forma a não comprometer o sistema como um todo ou a análise contínua do fluxo de dados entrante.

Sobre o estado atual dos estudos sobre o tema *concept drift* os autores em (LU et al., 2019) resumem sua análise com as seguintes percepções:

- Os métodos de detecção de *drift* baseados em análise de taxa de erro e na análise de distribuição de dados ainda representam um papel dominante na literatura, mas os métodos baseados em testes de hipóteses podem ser encontrados em artigos mais recentes;

- Sobre o processo de compreensão de *drift* (*drift understanding*), destacam que todos os métodos de detecção conseguem identificar "Quando" a mudança de conceito ocorre, mas muito poucos conseguem responder às questões sobre "Como" e "Onde" ocorrem;

- Modelos adaptativos e técnicas de *ensemble* têm representado um importante papel no processo de adaptação às mudanças. Em contrapartida, pesquisas sobre modelos de retreinamento com detecção explícita de mudança diminuíram;

- A maioria dos algoritmos de detecção e adaptação de *drift* assumem que o rótulo verdadeiro estará sempre disponível após a classificação ou predição. Poucas pesquisas têm sido direcionadas a métodos não-supervisionados ou semi-supervisionados de detecção e adaptação de *drifts*;

- Algumas técnicas de inteligência computacional, como lógica *Fuzzy*, por exemplo, têm sido aplicadas na análise de *concept drift*;

- Não há uma análise abrangente sobre fluxos de dados do mundo real a partir do aspecto de mudança de conceito, como o tempo de ocorrência da mudança, a gravidade da mudança e as regiões de mudança;

- Um número cada vez maior de pesquisas em outras áreas tem reconhecido a importância de lidar com a mudança de conceito, especialmente na comunidade de *big data*.

2.3.2 Características Gerais, Algoritmos e Outras Considerações

Em (BIFET; KIRKBY, 2009), os autores explicam que a quantidade cada vez maior de dados, gerada de forma contínua, é uma das motivações para a criação de algoritmos de mineração de fluxos de dados. Algoritmos tradicionais de mineração de dados, cujo processamento, em vários aspectos, é feito em lote, geralmente com a análise de todos os dados disponíveis, enfrentam dificuldades para lidar com o fluxo contínuo de dados. Isso é motivação para a busca por alternativas que sejam capazes de fazer o processamento de forma contínua, com a capacidade de fazer isso ocupando pouca memória, de maneira eficiente e capazes de gerar resultados competitivos com as técnicas tradicionais de mineração de dados. Em relação ao ciclo de processamento, (BIFET; KIRKBY, 2009) apresentam o ciclo de classificação de uma *stream* de dados, que inclui:

1. O algoritmo recebe o próximo registro da *stream* para processamento.

2. O algoritmo processa esse registro e atualiza suas estruturas, ou seja, atualiza seu modelo de aprendizagem. Faz isso utilizando pouco recurso de memória e da forma mais rápida possível.

3. O algoritmo está pronto para receber o próximo registro. Nesse estado, está pronto para fazer uma predição (no caso, uma classificação) para um registro ainda não avaliado, uma vez que o algoritmo já possui um modelo atualizado.

O ciclo apresentado para o processo de classificação, também é válido para algoritmos de regressão. De acordo com (BIFET et al., 2018) a regressão é uma tarefa de aprendizagem cujo objetivo principal é a predição de um valor numérico ao invés de um atributo categórico como na classificação. Para avaliar métodos de regressão, podem ser usadas praticamente as mesmas metodologias que as usadas em classificação como *holdout*, *prequential* e *i*nterleaved test-then-train. Excluem-se, nesse caso, as medidas de taxa de erro de classificação que não são aplicáveis a métodos de regressão.

Na lista a seguir estão indicados os algoritmos de regressão utilizados em experimentos preliminares deste estudo (descritos no Apêndice A):

- AdaGrad, (DUCHI; HAZAN; SINGER, 2011).
- AMRules Regressor, (ALMEIDA; FERREIRA; GAMA, 2013).
- ARF-Reg, (GOMES et al., 2018).
- FIMT-DD, (IKONOMOVSKA; GAMA; DzEROSKI, 2011).
- ORTO, (IKONOMOVSKA et al., 2011).
- Random AMRules, (DUARTE; GAMA, 2014).
- RandomRules (moa.classifiers.meta.RandomRules, (BIFET et al., 2010).
- FadingTargetMean (moa.classifiers.rules.functions.FadingTargetMean, (BIFET et al., 2010).

De acordo com (BIFET et al., 2018), o algoritmo FIMT-DD (acrônimo para *Fast Incremental Model Tree with Drift Detection*) é um algoritmo baseado em árvore de decisão para a regressão de fluxos de dados que é uma extensão do método Hoeffding Tree (DOMINGOS; HULTEN, 2000), mas que difere em alguns aspectos como no critério de divisão dos nós (*splitting criterion*), no tratamento de atributos numéricos, na poda (*pruning*), no uso de modelos lineares nas folhas, na forma de gerenciar mudanças de conceito e nas atualizações da árvore. Bifet et al. (2018) explica que, para a divisão dos nós, usa a redução da variância ao invés de ganho de informação. Para isso, o atributo escolhido é aquele em que a variância é maximamente reduzida se a árvore for dividida usando tal atributo. No caso de atributos numéricos, eles são manipulados usando uma extensão do método árvore binária exaustiva. Para a detecção de mudança de conceito, o teste de Page-Hinkley é usado nos nós internos da árvore de decisão para detectar mudanças na taxa de erro. Em relação à atualização da árvore, quando uma subárvore apresenta baixo desempenho, o FIMT-DD desenvolve uma árvore alternativa com novas instâncias de entrada e substitui a árvore original caso a nova apresente melhor desempenho. De acordo com Sobhy (2019) o FIMT-DD possui três parâmetros principais que afetam o seu erro de predição: o tipo de árvore adotada, onde uma árvore de regressão fornece melhores resultados do que uma árvore de modelo (*model tree*); o limite de empate, que é um limite abaixo do qual uma divisão será forçada a desfazer empates; e a taxa de aprendizagem, que é usada para treinar *perceptrons* nas folhas. Os autores em (GOMES et al., 2018) citam que o FIMT-DD também inclui um esquema de detecção de mudanças que sinaliza e adapta periodicamente as sub-ramificações da árvore onde aumentos significativos de variância são observados.

O AMRules Regressor, cujo nome pode ser interpretado como um modelo de regras adaptativo, é um algoritmo baseado em regras de decisão e estruturado de tal forma que

as regras do modelo podem ser adicionadas e removidas à medida que o processamento da *stream* evolui. Bifet et al. (2018) explica que, em relação à expansão de regras, de forma semelhante ao que ocorre no uso de Hoeffding Tree para árvores de decisão, o AMRules Regressor monitora, para cada regra e para cada atributo, a redução do desvio padrão (*standard deviation reduction*, SDR) que seria introduzida se o atributo fosse adicionado à regra. Se a proporção das duas maiores medidas de SDR entre todos os atributos potenciais exceder um limite, o recurso com o maior SDR é adicionado à regra para expandi-la. Quando um exemplo é recebido, ele é usado para atualizar as estatísticas para calcular os valores de SDR da primeira regra que o cobre (caso o conjunto de regras esteja ordenado), ou para calcular o SDR de todas as regras que o cobrem, se o conjunto estiver desordenado. Além disso, a expansão de uma regra só é considerada após ter recebido um certo número mínimo de exemplos. Em relação à criação de novas regras, se a regra padrão for expandida, ela se tornará uma regra normal e será adicionada ao conjunto de regras do modelo. Uma nova regra padrão é inicializada para substituir a expandida. Com relação à exclusão de regras, a taxa de erro de cada regra é monitorada com um teste de Page-Hinkley. Se o teste indicar que seu erro cumulativo excede um limite, a regra pode ser removida.

A autora em (SOBHY, 2019) explica que o método *FadingTargetMean*, baseado na média de *targets* de instâncias treinadas, usa um fator de esquecimento (*fading*) de forma a dar mais importância a dados mais recentes e menor importância a valores mais distantes das séries. Ele se baseia no pressuposto de que a importância dos exemplos diminui com a idade e permite que a abordagem se adapte às mudanças. O fator de esquecimento pode ser ajustado com base na necessidade de enfatizar a média do *target* mais recente *versus* a média de observações anteriores.

O algoritmo ARF-Reg (GOMES et al., 2018), corresponde a uma adaptação do método *Adaptive Random Forest* (ARF) (GOMES et al., 2017) para uso em problemas de regressão. Seus autores apresentam uma descrição do ARF-Reg em termos de sua estratégia de votação, indução de diversidade, características do algoritmo de aprendizagem básico (*base learner*) e dinâmica de atualização. Quanto à votação, citam que o algoritmo faz a média das previsões individuais para obter a previsão final; em relação à diversidade, explicam que ela é induzida na floresta treinando as árvores em diferentes subconjuntos de dados, limitando as decisões de divisão a um subconjunto de recursos selecionado aleatoriamente dos recursos de entrada originais; quanto ao *base learner*, citam que usa o FIMT-DD que, segundo os autores do ARF-Reg, possui um método eficiente de divisão e de seleção de atributos; sobre a dinâmica de atualização, explicam que o ARF-Reg depende de detectores de mudança de conceito internos e externos para cada árvore e do crescimento de árvores em segundo plano quando um aviso de mudança (*warning*) é detectado.

Os autores em (IKONOMOVSKA; GAMA; DžEROSKI, 2014) explicam que o

algoritmo ORTO (*Online Option Trees for Regression*) apresenta uma solução à dificuldade comum que algoritmos de árvore baseados na Hoeffding Tree apresentam no que se refere ao processo de divisão da árvore em casos ambíguos ou de atributos muito correlacionados. A estratégia desenvolvida consiste em usar nós de opção como um mecanismo de resolução dessa ambiguidade. Árvores de opções substituem a decisão de seleção de divisão padrão por uma divisão múltipla. Os autores explicam que, se pensarmos na indução de árvore como uma busca através do espaço de hipóteses H, um nó de opção é melhor entendido como um ponto de ramificação na trajetória de busca. Nesse caso, cada divisão opcional representa uma direção diferente que pode ser seguida na exploração do espaço de busca. E complementam que, quando várias divisões são permitidas, não há necessidade de esperar por mais evidências estatísticas para escolher entre as divisões com classificação mais alta. Ao explorar um subespaço maior de hipóteses H, espera-se que uma árvore de opções seja mais estável e mais robusta em comparação com uma árvore de decisão comum. Ao mesmo tempo, uma árvore de opções representa um conjunto compactado e conectado de árvores de regressão múltiplas. Além disso, complementam os autores em (IKONOMOVSKA; GAMA; DžEROSKI, 2014), que da mesma forma que o FIMT-DD, o algoritmo ORTO começa com uma folha vazia e lê os exemplos do fluxo na ordem de chegada. Cada exemplo é levado a uma folha onde as estatísticas necessárias são mantidas para cada ponto de divisão. Depois que um mínimo de n_{min} exemplos foram observados nas folhas da árvore, o algoritmo examina o critério de divisão. Se uma situação ambígua for encontrada, o algoritmo continuará criando um nó de opção. Caso contrário, um nó de divisão padrão será introduzido e o procedimento será executado recursivamente para os nós folhas nos quais um módulo de exemplos n_{min} foi observado. O algoritmo ORTO emprega duas estratégias diferentes para agregar várias previsões para um exemplo de teste: média não ponderada e a estratégia de escolha do melhor caminho. A estratégia de melhor caminho seleciona uma única previsão, considerada a melhor, obtida pela determinação do melhor caminho para uma folha em um determinado momento no tempo. De acordo com Gouk, Pfahringer e Frank (2019) FIMT-DD faz uso de modelos lineares nos nós folhas de cada árvore para aumentar a precisão das previsões, enquanto ORTO utiliza nós de opção para permitir que cada instância percorra múltiplos caminhos na árvore.

De acordo com os autores em (DUARTE; GAMA, 2014), o algoritmo Random AMRules faz uma combinação de conjuntos de regras e foi criado com a motivação de melhorar a performance do algoritmo AMRules ao adaptá-lo a uma estrutura de *ensemble*. Duarte e Gama (2014) explicam que modelos de regressão com um perfil de alta variação podem ser melhorados perturbando o conjunto de exemplos usados para treinamento. Por outro lado, modelos com perfil de baixa variância se beneficiam da perturbação do conjunto de atributos usados durante o treinamento do modelo. Ao observar que o AMRules tem um perfil de baixa variância, os autores do algoritmo Random AMRules tiveram a ideia de usar a estratégia implementada no algoritmo Random Forest a fim de tirar vantagem

da perturbação dos atributos e dos exemplos e, com isso, buscar reduzir o erro em tarefas de regressão.

O algoritmo AdaGrad (DUCHI; HAZAN; SINGER, 2011), ou *Adaptive Gradient Algorithm* (Algoritmo de Gradiente Adaptativo), é um algoritmo de otimização baseado no método de descida do gradiente. Ele é explicado com mais detalhes na próxima seção deste estudo.

Observa-se pelas breves descrições de alguns algoritmos, que as implementações diferem entre si pela abordagem básica que utilizam, uma vez que, alguns se baseiam em árvores de decisão, enquanto outros se baseiam em regras ou em outros modelos de implementação. Mesmo assim, alguns conceitos podem ser aplicados a todos em relação à forma como o processamento ocorre, podendo sofrer algumas adaptações dependendo da plataforma em que são executados.

O termo *prequential*, por exemplo, corresponde a uma abordagem de avaliação para a análise do processo de mineração de *stream* de dados que faz o processo de treino e de teste de forma contínua. De acordo com (BIFET et al., 2018), o *prequential* é um modelo de avaliação que aproveita o registro que está sendo avaliado, efetua um teste com ele e, após isso, atualiza o modelo de treinamento (*fitting*) do algoritmo. É similar ao modelo *Interleaved test-then-train*, também descrito em (BIFET et al., 2018), porém o *prequential* dá mais importância a registros mais recentes e faz uso de janela deslizante ou de um fator de decaimento. Outras abordagens de avaliação do processo de predição/classificação destacados por (BIFET et al., 2018), além do *prequential* e *interleaved test-then-train* incluem: *Holdout* (que é usado quando os dados são abundantes de tal forma que é possível separar conjuntos de testes obtidos periodicamente e medir a performance do modelo sobre esses dados) e o *Interleaved Chunks* (que segue o modelo do *Interleaved test-then-train*, mas com porções de dados em sequência, muitas vezes considerando porções de diferentes tamanhos).

Os autores em (GOMES et al., 2019) fazem uma análise do estado-da-arte do uso de técnicas de aprendizagem de máquina na mineração de fluxos de dados e citam desafios e oportunidades relacionados ao tema. Atividades que são típicas e bem estabelecidas para processos em *batch* apresentam dificuldades adicionais no universo da mineração de fluxos de dados. Isso ocorre pois um sistema de mineração de fluxos de dados precisa estar sempre disponível para processar os dados de entradas que chegam de forma contínua e devem fazer isso utilizando a menor quantidade de recursos (sejam recursos computacionais ou tempo) possível. Em relação ao pré-processamento dos dados, por exemplo, é preciso estabelecer técnicas adequadas para o tratamento de fluxos de dados quanto à transformação dos atributos (por exemplo: ajuste de escala, discretização dos dados), o tratamento de entradas inválidas, a redução de dimensionalidade e a seleção de *features*. Sobre o processo de aprendizagem indicam que a mineração de fluxos de dados requer atualização do modelo

em tempo real (ou próximo disso) e discutem sobre as relações entre fluxos de dados e séries temporais, a dificuldade de lidar com rótulos parcialmente conhecidos ou conhecidos com atraso, a aprendizagem com o uso de estratégias de conjuntos (*ensemble*), a necessidade de lidar com dados desbalanceados e com a detecção de anomalias.

Sobre o tema séries temporais, principal interesse deste estudo, Gomes et al. (2019) destaca que, diferentemente de um fluxo de dados regular (em que se assume que as instâncias estão distribuídas identicamente e de forma independente), os dados de uma série temporal geralmente apresentam forte dependência temporal. Essa característica demanda o estudo de técnicas especialmente adaptadas a esse cenário (como o uso de janelas de dados) para se construir um modelo de aprendizagem que leve em consideração as dependências existentes entre diferentes períodos de uma série. E isso não pode ser feito como num processo *batch* normal em que os dados da série são conhecidos previamente e é possível navegar pelos dados em vários sentidos a fim de indentificar as características inerentes à série de forma prévia ao seu processamento.

Para o processo de predição em séries temporais, os algoritmos de DSM aplicáveis são os da categoria algoritmos de Regressão. Um processo de regressão clássico, como a regressão linear, por exemplo, que é assunto da área de Estatística, geralmente é dependente da análise (em *batch*) de todos os dados históricos de uma série, para identificar a tendência e informações suficientes da série a fim de estimar um valor futuro, uma predição da série. Os autores em (BIFET et al., 2018) informam que esse é um problema difícil de computar incrementalmente, em especial no cenário da mineração de fluxos de dados; então, estratégias específicas para tratar a regressão em DSM devem ser buscadas, capazes de computar as informações de forma incremental e adaptativa.

Além dos processos de classificação e de regressão, os algoritmos DSM também estão presentes nas situações que demandam métodos de agrupamento (*Clustering*). Sobre esse modelo de aprendizagem de máquina, os autores em (BIFET et al., 2018) descrevem que *clustering* é uma tarefa de aprendizagem não-supervisionada que atua sobre dados que não contêm rótulos e consiste em distribuir um conjunto de instâncias em grupos de acordo com suas afinidades, ou suas similaridades. No cenário de DSM, algoritmos tradicionais para processos de agrupamento como o *k-means*, por exemplo, precisam ser adaptados, visto que devem passar a trabalhar com um fluxo contínuo de dados e não mais com um conjunto de dados disponível para processamento em *batch*. De acordo com (BIFET et al., 2018), converter o *k-means* para *streaming* não é uma tarefa trivial, visto que ele precisa analisar o conjunto de dados de forma iterativa e repetidamente. Por isso, muitos algoritmos de *clustering* para fluxos de dados têm duas fases: uma *on-line* e uma *off-line*. A fase *on-line* atualiza as estatísticas dos dados, produzindo uma forma de sumário dos dados, geralmente representado como um número reduzido de pontos chamados de *microclusters*. A fase *off-line*, então, usa esses sumários para computar um agrupamento

final de forma eficiente, fazendo isso em intervalos de tempo regulares ou sob demanda.

Os autores em (KREMPL et al., 2014) enumeram alguns desafios relacionados ao ciclo de descoberta de conhecimento (*kwowledge discovery*) na mineração de fluxos de dados e que podem ser elementos de pesquisa. Muitos desses desafios são comuns a processos *batch*, mas exigem atenção diferenciada no processamento de *stream* de dados. Tais desafios estão relacionados aos seguintes temas: a proteção de privacidade e confidencialidade; o manuseio de informação incompleta ou em atraso; o estudo de relações entre diferentes fluxos de dados; o desenvolvimento de métodos de detecção de eventos e modelos preditivos para dados censurados; o desenvolvimento de uma metodologia sistemática para o pré-processamento de *stream* de dados; a busca pela criação de modelos mais simples que considerem não apenas a precisão, mas também os recursos computacionais e a interpretabilidade; o estabelecimento de uma visão ampla em relação a avaliação de forma a lidar com a ausência de informações concretas sobre como os dados evoluem; a integração de sistemas de fluxos de dados com sistemas legados; e o desenvolvimento de sistemas de monitoramento online, garantindo a confiabilidade de atualizações e balanceamento da distribuição de recursos.

Sobre a proteção de privacidade e confidencialidade Krempl et al. (2014) destaca que é importante assegurar, por exemplo, que o modelo de aprendizagem não terá em sua estrutura informações incorporadas de forma desprotegida, que não garantam a proteção sobre a informação. Então, procedimentos de limpeza, ou descaracterização de atributos confidenciais que geralmente são feitos em processos *batch*, também devem ser avaliados para uso *on-line*. Citam que um desafio ao processar fluxos de dados é que o modelo é avaliado de forma gradual, incremental; ou seja, o modelo nunca é final, o que dificulta que se avalie se a privacidade está sendo garantida. Em relação ao pré-processamento, enquanto em processos *batch* as atividades de tratamento de dados redundantes ou faltantes, ou a eliminação de *outliers*, podem ser feitas previamente ao processamento dos dados, na mineração de fluxos de dados a eliminação de ruídos ou qualquer tipo de preparação dos dados deve ser feita à medida que a informação é processada.

Ainda em relação à mineração de fluxos de dados, é interessante destacar a existência de algumas abordagens híbridas na literatura, resultado de pesquisas em diferentes áreas para o processamento de fluxos contínuos de dados. Nesses casos, as abordagens incluem o uso de mecanismos de processamento de fluxos de dados que podem integrar, por exemplo, soluções de outra natureza, como algoritmos estatísticos adaptados para que mantenham seus modelos atualizados de forma frequente e automática. Um exemplo disso é apresentado por (ALBERG; LAST, 2018) que apresenta uma integração de modelos ARIMA (não sazonais e sazonais) com uma metodologia de aprendizagem incremental denominada de OLIN (*On Line Integration Network*), desenvolvida por Last (2002) para tarefas de classificação na presença de um *concept drift*. Os autores propõem um sistema ARIMA

incremental que tem como objetivo processar um fluxo contínuo de dados de entrada, como uma série de medições de carga em uma resolução de hora em hora ou diária. Ele reconstrói periodicamente o modelo preditivo usando a abordagem de janelas deslizantes para a seleção dos dados que serão utilizados como base para treinamento e validação e para a composição do modelo ARIMA (ou SARIMA). Utiliza a métrica MAPE (Mean Absolute Percentage Error) para selecionar o modelo mais adequado para a série em análise.

A opção pelo uso de algoritmos de *data stream mining* neste estudo, é apoiada pelo interesse em verificar a aplicabilidade de tais métodos no processo de *forecasting* de séries temporais, visto que as séries podem se apresentar como fluxos contínuos de dados. O monitoramento contínuo de consumo de recursos de um *datacenter*, por exemplo, pode ser representado como uma série temporal que registra a uma frequência constante os valores consumidos de espaço em disco, uso de processador ou espaço em memória disponível para a alocação de novas máquinas virtuais. Então, para prever os recursos necessários para tal *datacenter* para as próximas 24 horas, por exemplo, técnicas de predição cada vez mais otimizadas devem ser buscadas. O exemplo citado tem apenas caráter ilustrativo, mas serve para mostrar a aplicabilidade dos conceitos apresentados.

Na seção a seguir o algoritmo AdaGrad é descrito em maior detalhe por ter sido selecionado como o algoritmo DSM componente do método AA-ACF proposto por este estudo.

2.3.3 Algoritmo AdaGrad

AdaGrad (DUCHI; HAZAN; SINGER, 2011), ou *Adaptive Gradient Algorithm* (Algoritmo de Gradiente Adaptativo), é um algoritmo de otimização baseado no método de descida do gradiente, que tem por objetivo encontrar o mínimo de uma função (KARIM, 2018).

De acordo com (RUDER, 2017), AdaGrad adapta a taxa de aprendizagem de acordo com a frequência dos parâmetros, fazendo atualizações maiores para *features* pouco frequentes e atualizações menores para características que ocorrem com maior frequência. Ou seja, AdaGrad usa uma diferente taxa de aprendizagem para cada parâmetro (w_i), em cada passo de tempo t. Uma vantagem de AdaGrad em relação a outros algoritmos, destaca Ruder (2017), é a não necessidade de ajustar manualmente a taxa de aprendizagem.

A otimização, conforme (KARIM, 2018), pode ocorrer pela modificação do componente da taxa de aprendizagem (*learning rate*) e/ou pela modificação no componente de gradiente. No caso de AdaGrad, a otimização atua sobre a taxa de aprendizagem multiplicando um fator positivo a essa taxa de forma a tornar seu valor cada vez menor. Karim (2018) explica, também, que o algoritmo faz isso dividindo a taxa de aprendizagem pela raiz quadrada de S, que é a soma acumulada dos gradientes elevados ao quadrado

(gradientes atuais e passados, ou seja, até o tempo t). Dessa forma, Karim (2018) apresenta as Equações 2.18 e 2.19 a seguir:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t} \tag{2.18}$$

onde

$$S_t = S_{t-1} + \left[\frac{\partial L}{\partial w_t}\right]^2 \tag{2.19}$$

Nas equações, S (que corresponde à soma acumulada dos gradientes elevados ao quadrado) é inicializado com 0; t corresponde ao tempo (*time step*); w equivale ao peso/parâmetro a ser atualizado; α é a taxa de aprendizagem; e $\partial L/\partial w$ é o gradiente de L (que corresponde à função de perda, *Loss function*, ou função de custo, a minimizar com relação a w). Karim (2018) e Ruder (2017) explicam que a constante ϵ é adicionada como denominador a fim de evitar uma divisão por zero e pode ser definida como um valor bem pequeno (por exemplo: $\epsilon = 10^{-7}$).

De acordo com (RUDER, 2017), o algoritmo AdaGrad consegue trabalhar bem em situações com dados esparsos e em outros cenários, como no treinamento de redes neurais profundas de larga escala. Ruder (2017) cita um exemplo de uso do algoritmo no trabalho de (DEAN et al., 2012) em que AdaGrad é usado em conjunto com uma implementação de algoritmo SGD (*Stochastic Gradient Descent*) assíncrono, denominado de *Downpour SGD*, num problema de aprendizagem profunda distribuída de larga escala com mais de 1 bilhão de parâmetros. Isso mostra a capacidade do algoritmo em trabalhar com diferentes situações.

É importante mencionar, também, que o AdaGrad foi selecionado para compor o método AA-ACF proposto por este trabalho por seu desempenho no processamento de séries temporais mesmo em situações com baixo número de *features* e de registros, conforme pode ser observado nos experimentos descritos no Apêndice A.

2.4 Dependência Temporal

Os autores em (ŽLIOBAITė et al., 2015) citam que a dependência temporal também pode ser chamada de correlação serial ou autocorrelação e explicam que, em problemas de fluxo de dados, geralmente um conjunto de entrada de variáveis multidimensionais submetido a um algoritmo DSM contém as informações que tornam possível processar a classificação ou previsão dos dados. Ou seja, apenas os valores passados da variável dependente geralmente não são suficientes para o processamento por algoritmos dessa

natureza. Isso justifica o empenho na obtenção de informações adicionais sobre uma série temporal de forma a auxiliar o processo preditivo.

De acordo com Stojanova (2012), existem quatro tipos diferentes de autocorrelação: espacial, temporal, espaço-temporal e de rede (relacional). Stojanova explica que a autocorrelação espacial é definida como a correlação entre os valores dos dados que considera sua proximidade de localização. Nesta definição, as observações próximas são mais correlacionadas do que as distantes. A autocorrelação temporal refere-se à correlação de uma série temporal com seus próprios valores passados e futuros. Stojanova (2012) cita que também pode ser chamada de correlação defasada ou correlação serial, conforme sugerido por Žliobaitė et al. (2015). A autocorrelação espaço-temporal considera a correlação espaço-temporal entre as observações e a autocorrelação da rede expressa a interdependência entre os valores nos diferentes nós de uma rede.

O principal interesse deste estudo é explorar a autocorrelação temporal. A autora em (STOJANOVA, 2012) acrescenta que a autocorrelação temporal é a forma mais simples porque, considerando a complexidade da dimensionalidade, ela é geralmente de uma dimensão, no caso, o tempo. Em contraste, a autocorrelação espacial e de rede são mais complexas, geralmente aplicáveis a problemas multidimensionais. A autocorrelação é estudada em diferentes áreas de pesquisa.

Os autores em (NIELSEN; BRODTKORB; JENSEN, 2018) usam a autocorrelação para prever o movimento induzido por ondas de uma embarcação naval e o fazem combinando valores de função de autocorrelação e medições anteriores. Apesar de a autocorrelação poder expressar uma condição estacionária, os autores usam uma amostra de uma ACF (função de autocorrelação) obtida em uma janela de tempo recente para ajudar na previsão de um sistema dinâmico. Fazem isso considerando que os valores são válidos porque não mudaram significativamente. Outro ponto interessante que os autores citam em (NIELSEN; BRODTKORB; JENSEN, 2018) é que a função ACF pode ser vista como uma medida direta do efeito de memória de um processo físico. Eles argumentam sobre esse comportamento ao analisar as oscilações da superfície do mar; entretanto, neste trabalho, a ideia é experimentar esse conceito em diferentes conjuntos de séries temporais, de diferentes domínios.

Autores em (RODRIGUES; GAMA, 2009) usam coeficientes de autocorrelação em um estudo de previsão de fluxos de carga em estações de energia elétrica. Eles identificam a correlação de entradas históricas e usam os valores mais correlacionados como entrada para um sistema de filtro de Kalman usado em combinação com uma rede neural *perceptron* de várias camadas para criar novas previsões em diferentes horizontes, ou seja, para previsões antecipadas de carga para períodos de uma hora, um dia e uma semana.

Em (CAI et al., 2018) os autores exploram a dependência temporal no comportamento de grupos de usuários. Eles usam análise recorrente de eventos no contexto de

bancos de dados e usam janelas de tempo deslizantes e divisão de tempo na medição de comportamentos causais e dependentes. Os autores também sugerem que o uso de janela de tempo configurável pode ajudar a adaptar o sistema às diferentes necessidades.

Na referência (GUNAWARDANA; MEEK; XU, 2011) os autores apresentam um modelo para aprender dependências temporais em fluxos de eventos, chamado *Piecewise-Constant Conditional Intensity Model* (PCIM), que usa árvores de decisão para representar essas dependências.

Outro artigo que explora a dependência temporal é apresentado por Duong, Ramampiaro e Nørvåg (2018) que usa dependências temporais para detectar mudanças em fluxos de dados. Os autores apresentam um modelo denominado *Candidate Change Point Detector* (CCPD), usado para modelar dependências temporais de alta ordem para calcular as probabilidades de encontrar pontos de mudança no fluxo usando informações de dependência temporal de diferentes pontos do fluxo de dados.

Os autores em (HYNDMAN; ATHANASOPOULOS, 2018) explicam que, em um gráfico ACF, a relação r_k expressa entre as observações y_t e y_{t-k} de uma série pode ser escrita como na Equação 2.20:

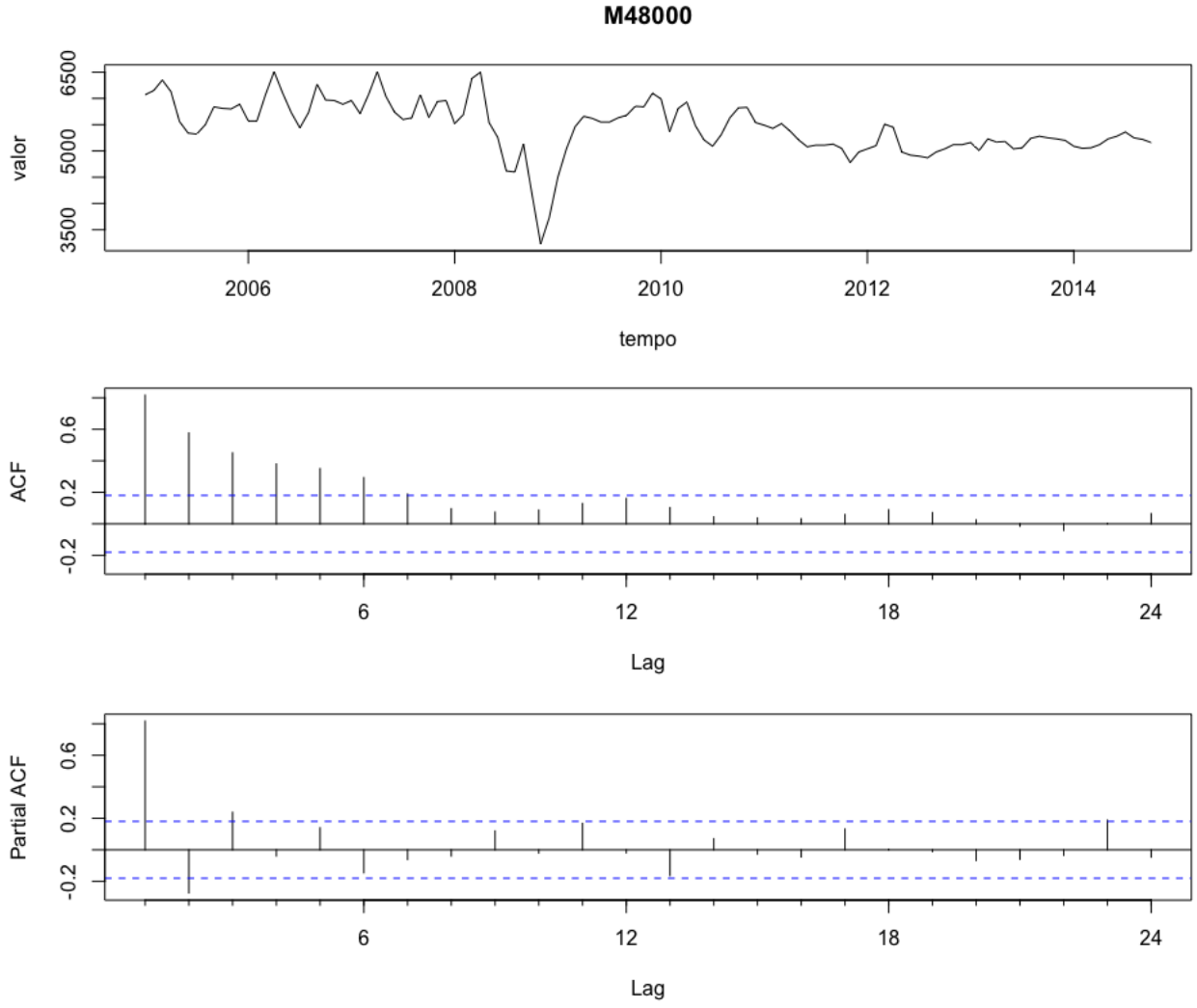
$$r_k = \frac{\sum_{t=k+1}^{N} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})^2} \tag{2.20}$$

onde k corresponde a um *lag* ou posição do coeficiente de autocorrelação, N representa a extensão da série temporal em análise e $\bar{y}$ a média de suas observações. Os autores informam que tendência e sazonalidade também podem ser avaliadas a partir da análise do gráfico ACF: dados com tendência tendem a ter valores positivos que diminuem lentamente à medida que as defasagens (*lags*) aumentam; dados sazonais, por outro lado, apresentarão maiores valores para os coeficientes de autocorrelação em múltiplos da frequência de sazonalidade. Segundo Werner e Ribeiro (2003), a função ACF pode auxiliar em testes de unidade para avaliar a estacionariedade de uma série. Em geral, ao analisar um gráfico ACF, é importante observar os coeficientes que apresentam valores mais significativos (estatisticamente diferentes de zero).

A Figura 11 apresenta os gráficos de autocorrelação (ACF) e autocorrelação parcial (PACF) para uma determinada série temporal do *dataset* M4. Nesses gráficos, os valores de cada coeficiente são calculados para cada *lag* (defasagem) da série e representados numa escala de valores que variam de 1 a -1. Para *lag=0*, os valores de coeficientes de autocorrelação e de autocorrelação parcial são iguais a 1 e indicam a correlação da observação inicial da série com ela mesma. Alguns autores, inclusive, não apresentam os coeficientes para *lag=0* devido a essa consideração. Além disso, é interessante observar que nos gráficos ACF e PACF existe uma faixa de valores destacada com uma linha tracejada, que representa o intervalo em que os valores não são estatisticamente relevantes.

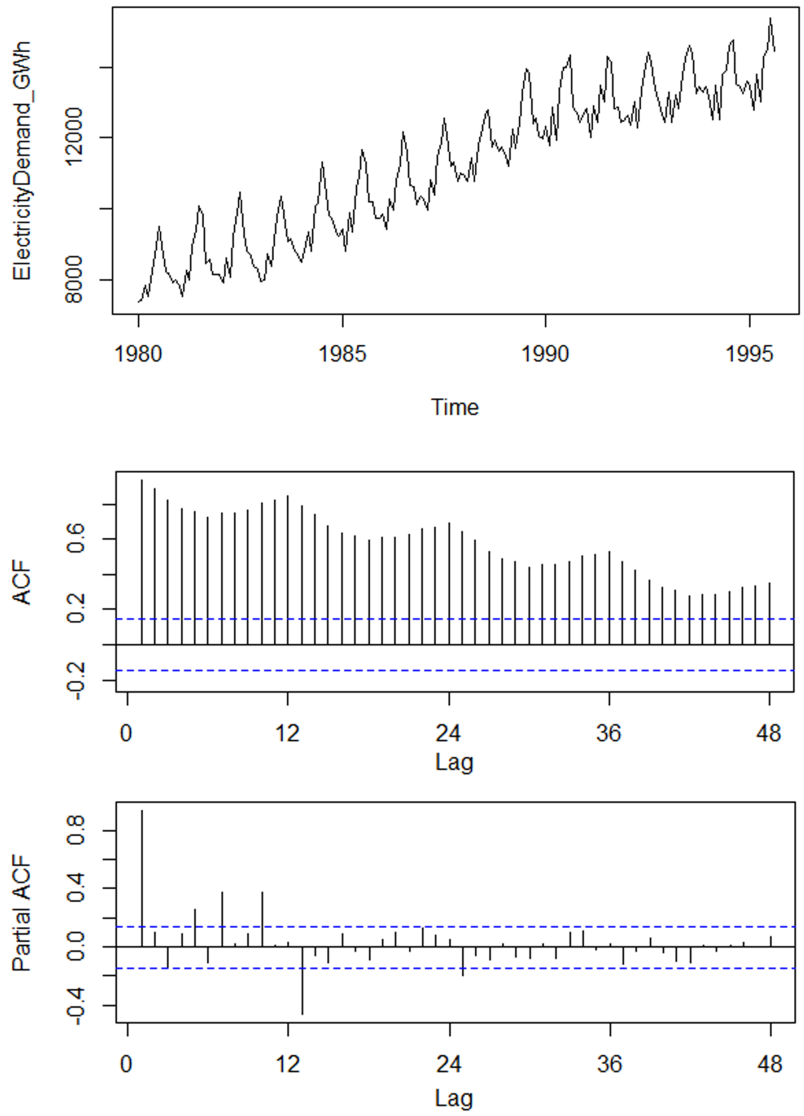
Na Figura 12 é apresentado o aspecto dos gráficos ACF e PACF para uma outra série. Pode-se observar que a série apresenta sazonalidade e tendência bem evidentes e essas características podem ser observadas pela forma como o gráfico ACF se comporta. A existência de tendência faz com que os dados do gráfico ACF decaiam de forma suave. A sazonalidade nesse exemplo é expressa pela forma ondulada como os coeficientes são apresentados no gráfico. Conforme (HYNDMAN; ATHANASOPOULOS, 2018), a autocorrelação parcial (PACF) apresentada nessa figura mede a relação entre y_t e y_{t-k} (da Equação 2.20 do ACF) depois de remover os efeitos dos *lags 1, 2, 3,..., k-1*.

Figura 11 – Gráficos de autocorrelação (*ACF plot*) e de autocorrelação parcial (*PACF plot*) da série M48000 de M4 (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a). Gráficos gerados com o uso das funções *Acf* e *Pacf* do pacote *forecast* (HYNDMAN et al., 2019b), na linguagem R.



Fonte: Autoria própria

Figura 12 – Exemplo de série com tendência e sazonalidade bem evidentes e seus respectivos gráficos ACF e PACF. Dados extraídos de dataset de demanda mensal de energia elétrica da Austrália (*Dataset elec* do pacote *fpp2* (HYNDMAN; ATHANASOPOULOS, 2018))



Fonte: Adaptado de exemplo disponível em (HYNDMAN; ATHANASOPOULOS, 2018)

O autor em (DATE, 2019) faz uma explicação didática sobre a autocorrelação parcial. Inicialmente, explica que uma série temporal autorregressiva pode ser expressa (de forma geral e com caráter simplificado) segundo a Equação 2.21:

$$T_i = \beta_0 + \beta_1 \times T_{(i-1)} \tag{2.21}$$

onde T_i corresponde ao valor que é predito pela equação para o *i-ésimo* período de tempo, β_0 indica o ponto em que o modelo intercepta o eixo Y e que aplica um viés constante

à previsão. β_1 indica a taxa na qual T_i muda em relação a $T_{(i-1)}$. O autor complementa que a principal suposição desse modelo é que a variância em $T_{(i-1)}$ é capaz de explicar as variâncias expressas por valores mais antigos que $T_{(i-1)}$, como se fosse capaz de capturar as informações associadas a valores mais antigos que ele. Só que isso nem sempre é verdade. Ou seja, pode ser que para obter o *forecast* para T_i seja necessário levar em conta $T_{(i-2)}$, gerando a necessidade de adaptar a Equação 2.21 para a Equação 2.22, introduzindo um novo termo referente ao *lag* $(i-2)$:

$$T_i = \beta_0 + \beta_1 \times T_{(i-1)} + \beta_2 \times T_{(i-2)} \tag{2.22}$$

Considerando isso, o autor em (DATE, 2019) apresenta como definição para autocorrelação parcial de uma série como a autocorrelação de T_i com uma versão defasada (*lagged*) dessa mesma série. Ou seja, $T_{(i-k)}$ é a correlação entre as seguintes variáveis: *Variável I*, que é o valor da variância de T_i que não é explicada pela variância em $T_{(i-1)}$, $T_{(i-2)}$,... $T_{(i-k+1)}$ e *Variável II*, que é o valor da variância em $T_{(i-k)}$ que não é explicada pela variância em $T_{(i-1)}$, $T_{(i-2)}$,...$T_{(i-k+1)}$. Como um exemplo, Date (2019) apresenta a Equação 2.23 para PACF para *lag=2*:

$$PACF(T_i, k=2) = \frac{Covariance(T_i \mid T_{(i-1)}, T_{(i-2)} \mid T_{(i-1)})}{\sigma_{T_i|T_{(i-1)}} \times \sigma_{T_{(i-2)}|T_{(i-1)}}} \tag{2.23}$$

onde o termo $T_i|T_{(i-1)}$ corresponde à série temporal dos resíduos obtidos pelo treinamento de um modelo linear para a distribuição de T_i versus $T_{(i-1)}$; o termo $T_{(i-2)}|T_{(i-1)}$, por sua vez, é a segunda série temporal dos resíduos que foram criados após o treinamento de um modelo linear para a distribuição de $T_{(i-2)}$ versus $T_{(i-1)}$. Nessa equação, o numerador calcula a covariância entre as duas séries temporais residuais e o denominador padroniza a covariância usando o desvio padrão de ambas as séries.

A Equação 2.23 apresenta uma expressão para $k = 2$ *(lag = 2)*. Uma expressão geral para PACF é obtida estendendo o raciocínio para *lags* superiores a 2.

Um quadro comparativo do comportamento dos gráficos ACF e PACF analisados em conjunto em relação a modelos de autorregressão (AR) e de médias móveis (MA, de *moving average*) é apresentado na Tabela 1. Analisar o comportamento desses gráficos é útil para estabelecer os coeficientes p e q em modelos autorregressivos como nos modelos ARMA e ARIMA, por exemplo.

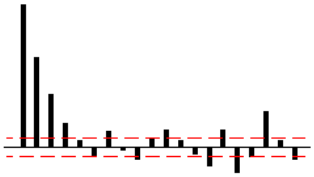
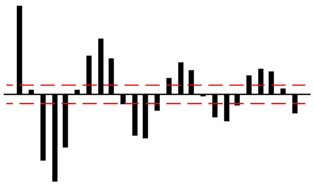
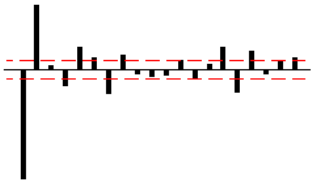
Outros quadros práticos são apresentados nas Tabelas 2 e 3 e sugerem como interpretar valores de ACF e PACF para auxiliar na identificação de parâmetros num modelo ARIMA. De acordo com o padrão observado, as tabelas sugerem avaliar também o gráfico PACF da mesma série. Importante observar que a análise deve ser feita com base nos *lags* para os quais os valores dos coeficientes estão além dos limites de significância.

Tabela 1 – Comportamento dos gráficos ACF e PACF analisados conjuntamente para modelos autorregressivos e de médias móveis.

Modelo	Comportamento de ACF	Comportamento de PACF
AR(p)	Diminui gradualmente	Decresce repentinamente após *p lags*
MA(q)	Decresce repentinamente após *q lags*	Diminui gradualmente
ARMA(p,q)	Diminui gradualmente	Diminui gradualmente

Fonte: Adaptado do original disponível em https://www.mathworks.com/help/econ/autocorrelation-and-partial-autocorrelation.html. Acesso em 28/09/2020.

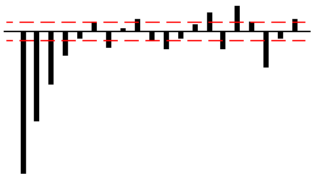
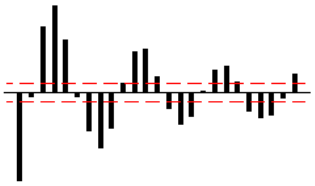
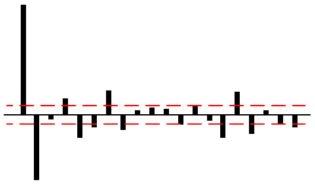
Tabela 2 – Quadro Auxiliar para Interpretação de Gráficos ACF para a Seleção de Parâmetros em Modelos ARIMA.

Padrão	Significado	Exemplo
Grande pico no *lag 1* que diminui depois de alguns *lags*.	Um termo autorregressivo nos dados. Usar a função de autocorrelação parcial para determinar a ordem do termo autorregressivo.	
Grande pico no *lag 1* seguido de uma onda decrescente que alterna entre correlações positivas e negativas.	Um termo autorregressivo de alta ordem nos dados. Usar a função de autocorrelação parcial para determinar a ordem do termo autorregressivo.	
Correlações significativas no primeiro ou segundo *lag*, seguidas por correlações que não são significativas	Um termo de média móvel nos dados. O número de correlações significativas indica a ordem do termo de média móvel.	

Fonte: Adaptado do original disponível em https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/time-series/how-to/autocorrelation/interpret-the-results/autocorrelation-function-acf/. Acesso em 28/09/2020.

Os autores em (FLORES; ENGEL; PINTO, 2012) usam funções de autocorrelação (ACF) e de autocorrelação parcial (PACF) para melhorar modelos de redes neurais (*Incremental Gaussian Mixture Network, IGMN,* e *Multilayer Perceptron, MLP*) na predição de séries temporais univariadas, com o objetivo de mostrar que as funções ACF e PACF podem ser usadas para reduzir o número de entradas de modelos neurais. De acordo com os autores, os coeficientes ACF são obtidos pela correlação linear de cada valor x_t da série com outros valores de diferentes *lags*, como x_{t-1}, x_{t-2} e assim por diante. Os coeficientes PACF, por sua vez, obtêm quase o mesmo resultado, mas removendo a interferência de

Tabela 3 – Quadro Auxiliar para Interpretação de Gráficos PACF para a Seleção de Parâmetros em Modelos ARIMA.

Padrão	Significado	Exemplo
Grande pico no *lag 1* que diminui depois de alguns *lags*.	Um termo de média móvel nos dados. Usar a função de autocorrelação para determinar a ordem do termo de média móvel.	
Grande pico no *lag 1* seguido de uma onda amortecida de valores que alternam entre correlações positivas e negativas.	Um termo de média móvel de alta ordem nos dados. Usar a função de autocorrelação para determinar a ordem do termo de média móvel.	
Correlações significativas no primeiro ou segundo *lag*, seguidas por correlações que não são significativas	Um termo autorregressivo nos dados. O número de correlações significativas indica a ordem do termo autorregressivo.	

Fonte: Adaptado do original disponível em https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/time-series/how-to/partial-autocorrelation/interpret-the-results/partial-autocorrelation-function-pacf/. Acesso em 28/09/2020.

outros valores. Citam no gráfico ACF a correlação entre x_t e x_{t-2} sobre a interferência de x_{t-1} e explicam que o PACF remove essa interferência. Os autores usam os coeficientes em dois momentos diferentes: no início da modelagem, para obter a janela de latência representativa de entrada e, após a modelagem, para analisar os resíduos.

Para este estudo, o foco está no uso de coeficientes de autocorrelação (ACF) na criação de atributos para auxiliar no processo de predição por algoritmos de mineração de fluxos de dados. Os resultados de experimentos prévios (disponíveis no Apêndice B), sugerem que o uso de tais coeficientes, ajustados para escalas de valores das observações das séries, auxiliam na obtenção de ganhos no processo de *forecast* em relação ao uso isolado de algoritmos estatísticos ou de DSM avaliados nos experimentos. O uso do conceito de dependência temporal nesse cenário é centrado em como traduzir aspectos temporais da série em atributos capazes de transformar séries univariadas em conjuntos de dados multivariados. No Capítulo 3 deste estudo, na seção de *Feature Engineering* será descrito como o conceito de dependência temporal será utilizado pelo método AA-ACF.

2.5 Métricas de Avaliação

Em problemas de aprendizagem de máquina é essencial a adoção de métricas para avaliar o desempenho de um processo de aprendizagem quanto à sua capacidade de processar dados de diferentes naturezas.

É preciso conseguir avaliar, por exemplo, num processo de classificação, quantas amostras de um conjunto de dados são classificadas corretamente e, num problema de predição, verificar o quão assertivo é determinado algoritmo. Sem dúvida, o tipo de informação envolvida, o volume de dados, o número de características que se conhece podem fazer com que um determinado processo apresente maior ou menor nível de acerto que outro.

Os autores em (SAIGAL; MEHROTRA, 2012) citam que os resultados obtidos com diferentes modelos de aprendizagem podem ser comparados de acordo com os seguintes critérios:

- Acurácia: a acurácia de um algoritmo de predição se refere a quão bem um dado preditor pode estimar o valor de um atributo (*target*) para momentos futuros ou para dados não vistos previamente.
- Velocidade: refere-se ao tempo necessário para a predição, mas principalmente aos custos computacionais envolvidos na geração e no uso de um determinado preditor.
- Robustez: corresponde à capacidade do preditor de fazer previsões corretas mesmo em situações com dados ruidosos ou dados com valores ausentes.
- Escalabilidade: refere-se à capacidade de construir um preditor que atue de forma eficiente mesmo em cenários com grandes quantidades de dados.
- Interpretabilidade: refere-se ao nível de compreensão e percepção que é fornecido pelo preditor.

Para o escopo dos experimentos realizados por este trabalho, o interesse está em avaliar medidas de acurácia de forma a identificar o quão preciso um método pode ser.

Em (HYNDMAN; KOEHLER, 2006) os autores comparam diferentes medidas de acurácia para a predição de séries temporais univariadas e propõem o uso de *Mean Absolute Scaled Error* (MASE) como uma medida padrão para comparar a acurácia de *forecasts* de múltiplas séries temporais. Os autores citam diferentes métricas existentes na literatura e que são usuais em competições relacionadas a séries temporais. Além disso, os autores dessa referência citam que em competições de predições de séries temporais (como as M-Competitions), também são usados *rankings* entre os métodos que participam da

competição e medidas baseadas no percentual de vezes que um método é melhor que um método estabelecido como *benchmark*.

Sobre as medidas de acurácia, Hyndman e Koehler (2006) citam as métricas dependentes de escala e sugerem que elas são úteis para comparar erros obtidos por diferentes métodos sobre os mesmo conjunto de dados, mas indicam que não deveriam ser utilizados para comparar erros de *datasets* que tenham diferentes escalas. Nessa categoria de métrica, citam o *Mean Square Error* (MSE, Equação 2.24), o *Root Mean Square Error* (RMSE, Equação 2.25), o *Mean Absolute Error* (MAE, Equação 2.26) e o *Median Absolute Error* (MdAR, Equação 2.27).

$$MSE = mean(e_t^2) \tag{2.24}$$

$$RMSE = \sqrt{MSE} \tag{2.25}$$

$$MAE = mean(\mid e_t \mid) \tag{2.26}$$

$$MdAE = median(\mid e_t \mid) \tag{2.27}$$

onde e_t é o erro residual dado por $Y_t - F_t$ (onde Y_t é o valor real de uma série para um determinado ponto de tempo t e F_t é o valor de *forecast*).

Outra categoria de medida inclui as baseadas em erros percentuais. De acordo com os autores em (HYNDMAN; KOEHLER, 2006), um erro percentual é definido pela equação $p_t = 100e_t/Y_t$. Observa-se na equação do erro percentual que ele é sujeito a gerar como resultado valores infinitos ou indefinidos se Y_t for igual a 0.

Nessa categoria de erros percentuais, Hyndman e Koehler (2006) citam: *Mean Absolute Percentage Error* (MAPE, Equação 2.28), *Symmetric Mean Absolute Percentage Error* (sMAPE, Equação 2.29), *Symmetric Median Absolute Percentage Error* (sMdAPE, Equação 2.30), *Median Absolute Percentage Error* (MdAPE, Equação 2.31), *Root Mean Square Percentage Error* (RMSPE, Equação 2.32) e o *Root Median Square Percentage Error* (RMdSPE, Equação 2.33). Em métricas como MAPE e MdAPE há, segundo os autores, a desvantagem de penalidade maior em erros positivos do que em erros negativos. Por essa razão, métricas como sMAPE e sMdAPE foram criadas com a denominação de medidas "simétricas" de forma a evitar esse tipo de viés, eliminando a possibilidade de gerar erros com valores negativos. No entanto, apesar de serem chamados de simétricos, a simetria provavelmente só ocorre pelo fato de que a ordem dos valores reais e preditos na fórmula pode ser trocada e o resultado apresentado é o mesmo devido ao uso do valor absoluto da diferença entre os termos do numerador das equações. O fato da simetria

em relação a evitar o viés de erros negativos, por sua vez, é facilmente contestada pelo seguinte exemplo: considerando $Y_t = 100$ e $F_t = 150$, $sMAPE = 0,4$. Para $Y_t = 100$ e $F_t = 50$, $sMAPE = 0,67$.

$$MAPE = mean(\mid p_t \mid) \tag{2.28}$$

$$sMAPE = mean(200 \mid Y_t - F_t \mid /(Y_t + F_t)) \tag{2.29}$$

$$sMdAPE = median(200 \mid Y_t - F_t \mid /(Y_t + F_t)) \tag{2.30}$$

$$MdAPE = median(\mid p_t \mid) \tag{2.31}$$

$$RMSPE = \sqrt{mean(p_t^2)} \tag{2.32}$$

$$RMdSPE = \sqrt{median(p_t^2)} \tag{2.33}$$

Sobre medidas baseadas em erros relativos, Hyndman e Koehler (2006) explicam que nessas medidas os erros obtidos são divididos pelos erros obtidos pelo uso de um método padrão de *forecasting*. O erro relativo pode ser obtido pela expressão $r_t = e_t/e_t^*$, onde e_t^* corresponde ao *forecast* gerado pelo método de referência (que pode ser, por exemplo, o *random walk* ou outro método). Nessa categoria, incluem os erros *Mean Relative Absolute Error* (MRAE, Equação 2.34), *Median Relative Absolute Error* (MdRAE, Equação 2.35) e *Geometric Mean Relative Absolute Error* (GMRAE, Equação 2.36).

$$MRAE = mean(\mid r_t \mid) \tag{2.34}$$

$$MdRAE = median(\mid r_t \mid) \tag{2.35}$$

$$GMRAE = gmean(\mid r_t \mid) \tag{2.36}$$

Outra categoria de medida são os erros escalados. De acordo com Hyndman e Koehler (2006) o erro é escalado com base numa amostra de erro absoluto médio (MAE) calculado com forecasts obtidos com o método naïve (no caso, random walk). O erro

escalado pode ser definido pela Equação 2.37. Nessa categoria, destacam o Mean Absolute Scaled Error (MASE, Equação 2.38).

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n} \mid Y_i - Y_{i-1} \mid} \tag{2.37}$$

$$MASE = mean(\mid q_t \mid) \tag{2.38}$$

Nas competições M3 e M4 a métrica sMAPE foi utilizada como uma das métricas para a comparação de desempenho dos diferentes métodos de predição propostos pelos competidores.

Para a competição M5 (https://mofc.unic.ac.cy/m5-competition/), por sua vez, a acurácia dos *forecasts* pontuais foi avaliada usando o *Root Mean Squared Scaled Error* (RMSSE), que é (de acordo com as diretrizes da competição) uma variante do *Mean Absolute Scaled Error* (MASE) proposto por (HYNDMAN; KOEHLER, 2006). A Equação 2.39 mostra como RMSSE é calculado.

$$RMSSE = \sqrt{\frac{1}{h}\frac{\sum_{t=n+1}^{n+h}\left(Y_t - \hat{Y}_t\right)^2}{\frac{1}{n-1}\sum_{t=2}^{n}\left(Y_t - Y_{t-1}\right)^2}} \tag{2.39}$$

onde Y_t é o valor real do dado de teste no ponto t, $\hat{Y}_t$ é o *forecast* gerado pelo método, n é o comprimento da amostra de treinamento (número de observações históricas) e h é o horizonte de predição.

Os autores da competição M5 justificam a escolha pelo uso de RMSSE pelos seguintes fatores (conforme descrito em manual com diretrizes da competição disponível em https://mofc.unic.ac.cy/wp-content/uploads/2020/03/M5-Competitors-Guide-Final-10-March-2020.docx):

- As séries de M5 são caracterizadas por intermitência, envolvendo muitos zeros. Isso significa que métricas de erros absolutos (que são otimizados para a mediana) iriam atribuir melhor pontuação a métodos que gerassem *forecasts* próximos de zero. O RMSSE, por sua vez é otimizado para a média, pois o principal interesse de M5 é avaliar o erro médio.
- A medida é independente da escala, o que significa que pode ser usada com eficácia para comparar previsões em séries com escalas diferentes.
- Diferentemente de outras métricas que são sensíveis a valores zerados, o RMSSE pode ser calculado com segurança, pois não depende de divisões com valores que podem ser iguais a zero (que poderiam gerar indeterminações).

- A medida penaliza igualmente erros de previsão positivos e negativos, bem como previsões grandes e pequenas, podendo portanto ser considerada uma métrica simétrica.

Os autores em (CHAI; DRAXLER, 2014) citam que um importante aspecto da métrica usada para avaliar um modelo é a sua capacidade de discriminar entre os resultados apresentados por diferentes modelos. Sugerem que a medida usada deve ser capaz de produzir grandes variações nas suas medidas para diferentes conjuntos e modelos. Além disso, explicam que ao avaliar diferentes modelos usando uma única métrica, é importante avaliar as diferenças existentes nas distribuições dos erros.

Com base nas métricas citadas, é possível identificar que várias são as medidas de acurácia aplicadas ao processo de predição de séries temporais, todas elas possíveis de serem utilizadas. Portanto, compete avaliar em cada estudo (de acordo com o conjunto de séries envolvidas e com o tipo de análise comparativa que se deseja fazer) qual a medida mais ideal a ser selecionada.

Para o interesse deste estudo, a métrica selecionada é o sMAPE, utilizado pelas competições M3 e M4 cujos *datasets* somados correspondem a mais de 72% do total de séries que serão utilizadas na etapa de Experimentos deste trabalho. Portanto, o uso dessa métrica possibilita avaliar o desempenho do método proposto com outros valores de referência.

2.6 Considerações Finais

O desenvolvimento de um trabalho de pesquisa envolve o conhecimento de conceitos básicos acerca dos temas correlacionados à área de estudo e o acesso a obras de outros autores que sejam relevantes para o objetivo que se busca alcançar com a pesquisa.

Neste capítulo foram apresentados conceitos gerais sobre os temas Séries Temporais, Multisséries Temporais, Mineração de Fluxos de Dados, Dependência Temporal e Métricas de Avaliação com o objetivo de introduzir ao leitor deste estudo informações básicas importantes acerca dos temas discutidos no trabalho e apresentar referências importantes utilizadas como base para o desenvolvimento dos experimentos realizados ao longo do estudo.

No próximo capítulo, será descrito o método que é objetivo de pesquisa deste estudo. Inicialmente, serão apresentadas as tecnologias utilizadas e atividades que o compõem e, em seguida, será feita uma demonstração de uso do método.

3 Método

3.1 Considerações Iniciais

Este capítulo tem o objetivo de descrever um método de predição de multisséries temporais univariadas que faz o uso combinado de método estatístico e algoritmo de mineração de fluxo de dados. O método tem como uma de suas proposições principais explorar o conceito de dependência temporal em seu processo de *feature engineering* (engenharia de atributos). Com a estrutura final estabelecida para o método, propõe-se um modelo capaz de atingir resultados competitivos com potencial de superar os resultados individuais apresentados pelos algoritmos envolvidos.

Um método exploratório de trabalho foi utilizado para chegar à proposição final do método apresentado neste capítulo. Os experimentos prévios que levaram à estrutura proposta para o método são descritos nos Apêndices A e B, que apresentam, respectivamente, as análises efetuadas para se estabelecer o algoritmo de mineração de fluxos de dados a utilizar no método e os parâmetros a utilizar no processo de *feature engineering* baseado em coeficientes de autocorrelação.

Os conceitos e as tecnologias apresentados na seção de Revisão de Literatura são levados em consideração durante todo o processo de desenvolvimento do método.

A seguir, o método é descrito em detalhes, indicando as estruturas de dados envolvidas, os algoritmos utilizados, o processo de seleção de algoritmo e a estratégia de combinação proposta para a obtenção da predição final de cada série.

3.2 Descrição do Método

O diagrama da Figura 13 apresenta um esquema do método de predição proposto, e apresenta as atividades realizadas para a predição de uma série termporal. O método é identificado com a sigla **AA-ACF**, como indicação dos principais métodos e conceitos que foram considerados em seu desenvolvimento (o primeiro 'A' como acrônimo para auto.arima, algoritmo estatístico do método; o segundo 'A' representando AdaGrad, algoritmo DSM do método; e 'ACF' indicando que faz uso de coeficientes de autocorrelação em sua estrutura). Na comparação com diferentes técnicas de seleção utilizadas na seção de experimentos deste documento, o método pode ser citado como método de **Seleção e Fusão**, correspondente à configuração *default* do método que prevê a realização de fusão de *forecasts* de auto.arima e AdaGrad nos casos em que o algoritmo selecionado para a série em análise seja o AdaGrad. Uma variante do método apresentada no mesmo

diagrama, corresponde à versão em que o usuário pode optar por não utilizar o processo de fusão; nesse caso, o método será identificado como método de **Seleção NRegs**, por simplesmente fazer a seleção de algoritmo mais indicado para a série (estatístico ou DSM) com base no erro de treinamento calculado com base em n registros finais de treinamento da série.

Usando uma descrição resumida, **o método faz uso do conceito de dependência temporal no processo de *feature engineering*, seleciona o algoritmo mais adequado para a série com base na medição de erro de treinamento e realiza a predição para a série temporal em análise usando um método estatístico clássico e/ou um algoritmo DSM de acordo com o resultado da seleção**.

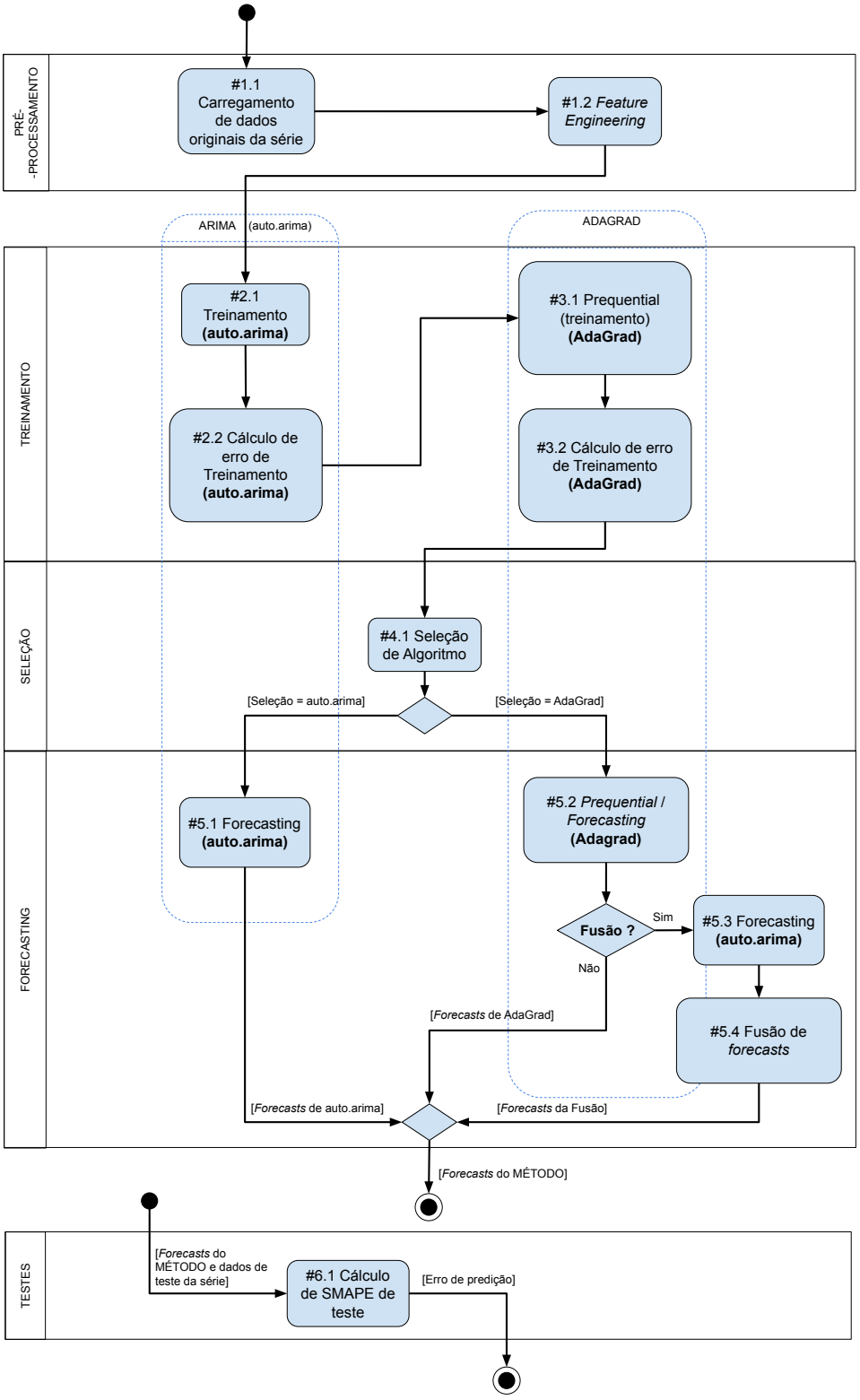
Apesar de o processamento ser realizado por série, é importante destacar a sua classificação como um método para processamento multissérie por ter sido elaborado a partir da análise de seu comportamento no *forecast* de milhares de séries temporais em conjunto. Além disso, o método busca minimizar o erro de predição individual ao selecionar o algoritmo de predição mais adequado para cada série.

As atividades do método estão organizadas em etapas de Pré-processamento, Treinamento, Seleção e *Forecasting*. O diagrama do método inclui, também, a atividade de Testes, que corresponde à verificação de acurácia do processo de predição.

Na etapa de Pré-processamento, os dados da série são carregados e preparados para o processo de predição. Na etapa de Treinamento, os dados das séries são submetidos para treinamento com os algoritmos auto.arima e AdaGrad; nessa etapa o erro de treinamento é calculado para cada algoritmo e é tomado como base na etapa seguinte, que é a etapa de Seleção, que indicará qual o algoritmo mais indicado para o processo de predição. Na etapa de *Forecasting*, são geradas as predições para a série utilizando o algoritmo selecionado na etapa anterior. No caso de o processo de seleção indicar AdaGrad, os valores de *forecasting* podem passar por um processo de fusão para definir os valores de predição finais da série. Conforme comentado anteriormente, o processo de fusão pode ser executado de acordo com o parâmetro de entrada escolhido pelo usuário (que indica se o método deve utilizar ou não o método de fusão). Todavia, como método principal para avaliação de desempenho, será utilizada a versão com **Seleção e Fusão** (configuração *default*), que terá seu desempenho comparado com a variante **Seleção NRegs** e outras técnicas alternativas na seção de Experimentos e Resultados deste estudo.

Com o objetivo de explicar o processo de forma mais detalhada, as atividades que compõem o método estão descritas nos itens a seguir nesta seção.

Figura 13 – Diagrama do método de predição AA-ACF.

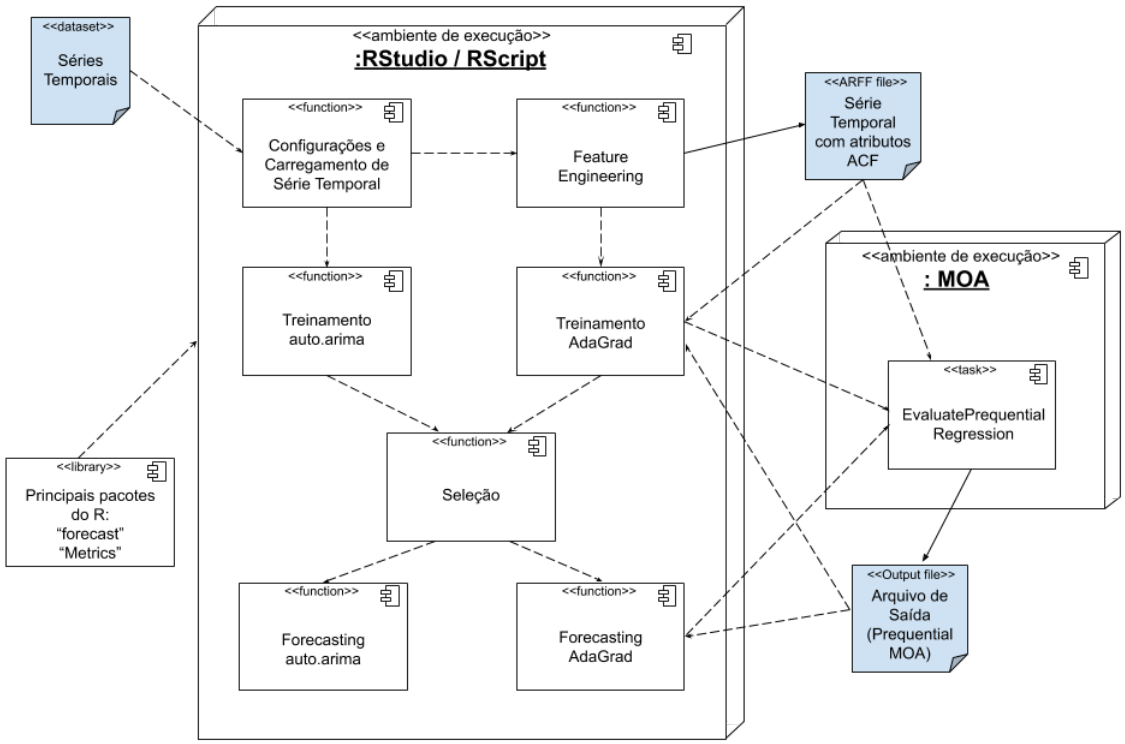


Fonte: Autoria própria

3.2.1 Tecnologias e Algoritmos Utilizados

A Figura 14 apresenta um diagrama de componentes simplificado com os principais elementos que compõem a aplicação criada para experimentos com o método AA-ACF. De uma forma geral, um programa em linguagem R controla o carregamento das séries temporais de um *dataset* e a sequência de atividades necessárias para o processamento de cada série. O software MOA[1] (BIFET et al., 2010) é executado a partir de uma chamada no código escrito em R e gera um arquivo de saída (em texto) com os resultados do treinamento/predição da série no que se refere à predição com algoritmos DSM, no caso o AdaGrad.

Figura 14 – Diagrama de componentes do método AA-ACF.



Fonte: Autoria própria

Para a preparação de dados, o método utiliza programas em linguagem R (IDE RStudio) para o processo de *feature engineering* e geração de arquivos ARFF (*Attribute-Relation File Format*) para o processamento no *software* de mineração de fluxos de dados MOA.

[1] MOA, acrônimo de *Massive Online Analysis,* é um *framework open-source* de mineração de fluxos de dados, que inclui algoritmos de classificação, regressão, agrupamento, detecção de *outliers*, detecção de mudança de conceito e sistemas de recomendação, escrito em Java e disponível em https://moa.cms.waikato.ac.nz/

Para a predição de séries usando o algoritmo ARIMA, o método utiliza a função *auto.arima*, do pacote *forecast* (HYNDMAN et al., 2019b) do R, por esse ser um método que, automaticamente, procura estabelecer os parâmetros ideais para a execução do método de regressão ARIMA.

Para a predição de séries usando o algoritmo AdaGrad, um programa em linguagem R executa, de forma combinada, a criação de arquivos ARFF com os atributos adicionais selecionados para a série e realiza a predição de AdaGrad pela chamada de execução do software MOA a partir do código em R. Os arquivos de saída gerados pelo MOA em formato texto são considerados nesse processo.

O *ensemble* (a combinação de diferentes algoritmos) é obtido pelo uso de códigos em linguagem R. Após a seleção do algoritmo (com base no erro de treinamento obtido), o processo de *forecasting* é realizado. No caso de seleção de auto.arima, as predições feitas por esse algoritmo correspondem aos *forecasts* finais da série. Se for selecionado AdaGrad, o método calcula os *forecasts* para esse algoritmo e depois faz (opcionalmente, conforme selecionado pelo usuário) um processo de fusão dos valores preditos com os valores gerados por auto.arima. A fusão é feita pelo cálculo de uma média simples entre os *forecasts* dos dois algoritmos.

A escolha de AdaGrad como algoritmo de mineração de fluxos de dados utilizado pelo método é justificada pelo seu desempenho superior em relação aos demais algoritmos de DSM avaliados. Isso fica evidenciado pelos resultados obtidos durante os experimentos realizados na Fase 1 de experimentos (ver Apêndice A), em especial pelos resultados obtidos a partir do Experimento 1.6. Outro ponto importante a destacar é que o uso combinado de técnicas distintas, que justificam a opção pelo uso conjunto de auto.arima e AdaGrad, podem ser evidenciados a partir do Experimento 1.8 (também detalhado no Apêndice A).

A Figura 15 apresenta um algoritmo geral do funcionamento do método AA-ACF. No algoritmo estão apresentadas em pseudocódigo (com uma descrição textual simplificada) as principais atividades relacionadas ao processamento de todas as séries de um determinado *dataset*.

3.2.2 Atividades da Etapa de PRÉ-PROCESSAMENTO

As atividades da etapa de Pré-processamento são responsáveis pelo carregamento de dados das séries temporais a serem processadas pelo método e pela execução do processo de *feature engineering*, necessário para o processamento de séries com algoritmos DSM.

Figura 15 – Algoritmo do método AA-ACF

```
Algoritmo – Método AA-ACF
1:  entrada: datasetDeSeriesTemporais, parametrosDeProcessamento, usaFusao (default=TRUE)
2:  saída: arquivoForecastsErros, arquivoConsolidadosPorPeriodicidade
3:  início_método
        // carrega dataset ou referência ao dataset a ser processado
4:      dataset ← datasetDeSeriesTemporais
        // identifica parâmetros globais para processamento do dataset como a lista de séries a serem
        // processadas, a periodicidade das séries (anual, trimestral, mensal, diária, horária,...)
        // e o horizonte de predição a ser considerado para as séries de cada periodicidade.
5:      parametrosGlobais ← parametrosDeProcessamento
        // processa as séries do dataset
6:      para cada periodicidade a ser processada faça:
7:              listaDeSeries ← lista de séries a serem processadas referentes à periodicidade
                // percorre a lista de séries da periodicidade selecionada
8:              para cada série temporal da listaDeSeries faça:
                        // carrega dados de uma série temporal do dataset
9:                      S ← observações da série temporal
                        // cria atributos adicionais da série
10:                     ARFFcomFeaturesACF ← featureEngineering (S)
                        // realiza o treinamento com auto.arima e identifica erro de treinamento
11:                     erroTreinamentoAutoArima ← treinoECalculoErroAutoArima(S)
                        // realiza o treinamento com AdaGrad e identifica erro de treinamento
12:                     erroTreinamentoAdaGrad ← prequentialTreinoECalculoErroAdaGrad(ARFFFeaturesACF)
                        // realiza a seleção de algoritmo comparando os erros de treinamento
13:                     se (erroTreinamentoAdaGrad < erroTreinamentoAutoArima) então:
                                // processa forecasts com método AdaGrad
14:                             forecastsAdaGrad ← prequentialForecastingAdaGrad(ARFFcomFeaturesACF)
15:                             se (usaFusao) então:
16:                                 forecastsAutoArima ← forecastingAutoArima(S)
17:                                 forecastsAdaGrad ← prequentialForecastingAdaGrad(ARFFcomFeaturesACF)
                                    // faz a fusão dos forecasts de auto.arima e AdaGrad para definir os
                                    // forecasts da série. A fusão corresponde a uma média simples dos valores.
18:                                 forecastsSerie ← média(forecastsAutoArima, forecastsAdaGrad)
19:                             senão forecastsSerie ← forecastsAdaGrad
20:                             fim_se
21:                     senão forecastsSerie ← forecastingAutoArima(S)
22:                     fim_se
                        // computa erro de teste, caso dados de teste estejam disponíveis
23:                     erroTeste ← smape(dadosTeste, forecastsSerie)
                        // grava forecasts da serie e erros de teste (sMAPE para cada horizonte de predição)
24:                     gravaForecastsSerieEmArquivoExterno()
25:             fim_para
                // consolida dados (erros médios) das séries da periodicidade e grava em arquivo
26:             gravaDadosPeriodicidadeEmArquivoExterno()
27:     fim_para
28: fim_método
```

Fonte: Autoria própria

Atividade #1.1 - Carregamento de dados originais da série

O processo de carregamento de dados originais da série tem como finalidade importar para o método os dados de treinamento e de teste da série. De uma forma geral, é suficiente para o método receber um vetor de valores numéricos representando cada uma

dessas porções de dados (treino e testes). Os dados de testes não são obrigatórios, visto que num processo de predição real nem sempre eles estariam disponíveis. Mas, para fim de avaliação de acurácia, serão considerados neste estudo como uma porção dos dados associados a cada série.

Esse processo de carga de dados deve ser ajustado de acordo com o *dataset* em análise, uma vez que a fonte de dados pode estar armazenada de diferentes maneiras, em um banco de dados externo, num pacote de dados (como num *package* da linguagem R), num conjunto de arquivos CSV, em arquivos textos, em arquivos ARFF, entre outros formatos válidos.

Nessa etapa, além de se estabelecer como fazer o acesso e leitura dos dados das séries, é preciso inicializar parâmetros que definem se o processo de fusão será utilizado ou não, e qual o horizonte de predição que deverá ser usado para cada conjunto de séries. Esses parâmetros relativos ao horizonte de predição dependem, especialmente, da periodicidade das séries e da quantidade de *forecasts* que se deseja para cada *dataset*. Tais parâmetros definem, também, como estabelecer um código identificador para cada série e como carregar separadamente os dados de testes das séries. A Tabela 4 apresenta os horizontes de predição convencionados para este estudo.

Tabela 4 – Horizontes de predição estabelecidos de acordo com a periodicidade das séries. Utilizados como referência os horizontes de predição estabelecidos para a competição M4.

Periodicidade	**Horizonte de predição (h)**
Séries horárias (*hourly*)	48 *forecasts*
Séries diárias (*daily*)	14 *forecasts*
Séries semanais (*weekly*)	13 *forecasts*
Séries mensais (*monthly*)	18 *forecasts*
Séries trimestrais (*quarterly*)	8 *forecasts*
Séries anuais (*yearly*)	6 *forecasts*

Fonte: Autoria própria

Para as predições com o método estatístico ARIMA (função *auto.arima*) são utilizadas estruturas de dados contendo, unicamente, a sequência de observações da série, adaptadas para um tipo de dado denominado na linguagem R de *time series*. Para isso, os dados das séries são carregados para o ambiente do método e as informações de interesse da série são selecionadas, no caso, o seu código identificador (se tiver, senão um número sequencial pode ser atribuído para a série) e as observações (eventos da série). O método proposto dispensa a necessidade de referência temporal (dia, mês, ano, bimestre,... ao qual a série se refere), pois nem sempre essa informação está presente no conjunto de dados originais da série.

Para as predições com o algoritmo de mineração de fluxos de dados AdaGrad, por sua vez, a série deve passar por um processo de *feature engineering*, descrito na atividade a seguir.

Atividade #1.2 - *Feature Engineering*

O processo de *feature engineering* (engenharia de atributos) tem a finalidade de criar atributos que serão adicionados às características originais da série com o objetivo de fornecer mais informações para o processo de treinamento realizado pelos algoritmos de aprendizagem.

Uma característica importante do método é a utilização do conceito de dependência temporal na criação de atributos para a série. Para isso, coeficientes obtidos com a função de autocorrelação (ACF) são extraídos da série e usados na criação de novos atributos.

Considerando a natureza dos algoritmos DSM, que não analisam previamente todos os dados em lote (*batch*), mas trabalham com um processamento incremental dos dados, foi utilizado o conceito de janela deslizante. O uso dessa abordagem de janela deslizante permite ajustar os coeficientes de autocorrelação durante todo o processo de criação dos atributos, adaptando-os ao aspecto mais recente dos eventos da série.

Os atributos adicionais associados a cada observação da série são criados com base em eventos anteriores (valores de observações anteriores da série), usando janelas deslizantes de até 288 observações (referidas neste estudo como W288). Os dados da janela são usados para calcular os coeficientes ACF que expressam a relação entre uma observação e os eventos que a precedem.

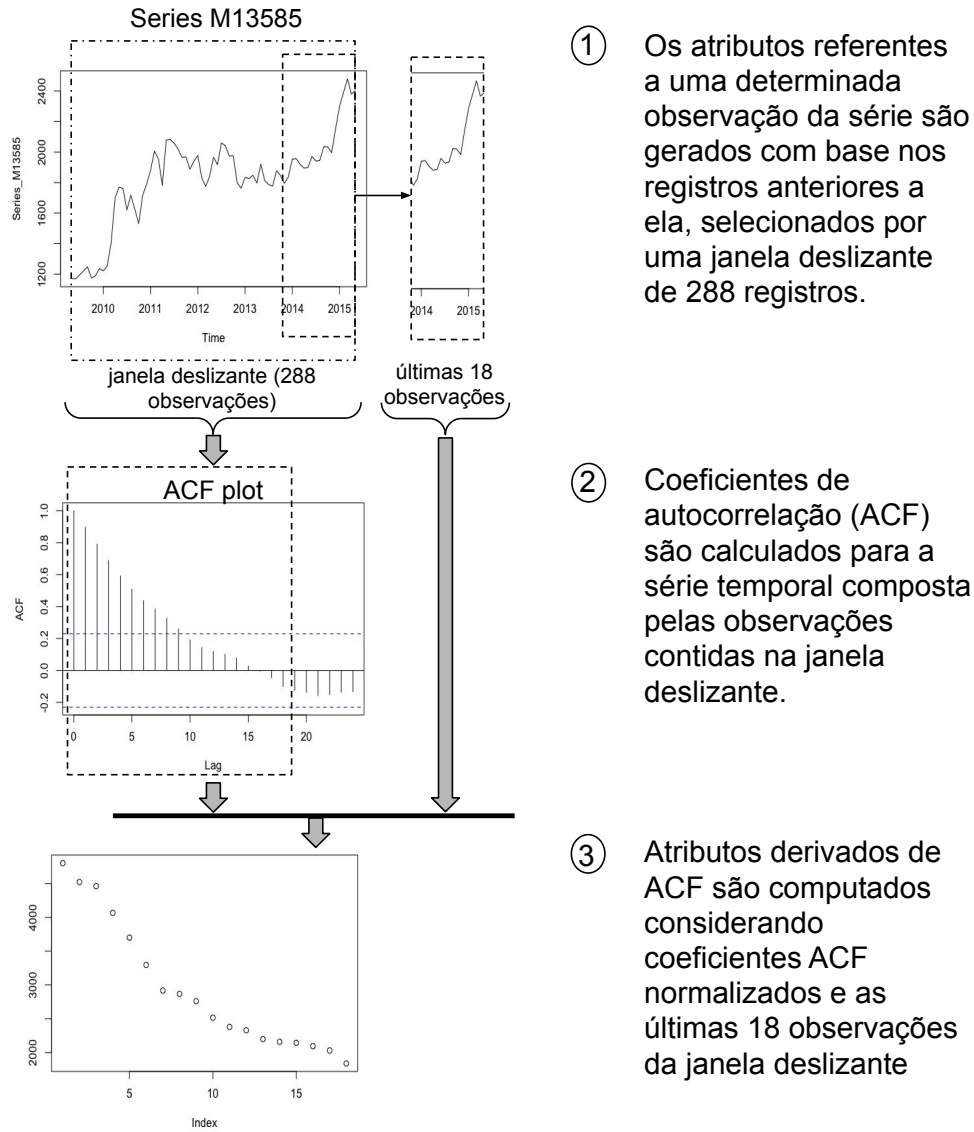
O método considera os primeiros 18 coeficientes de autocorrelação (*lags* 1 a 18) como sendo os mais relevantes para o propósito desta abordagem, independente de seus valores. Os valores positivos e negativos são considerados como válidos, sem necessidade de verificar se estão dentro do limite de significância.

O raciocínio por trás do uso de atributos de dependência temporal no processo de engenharia de atributos foi baseado no estudo de Žliobaitė et al. (2015), que propõe o uso de autocorrelação temporal em problemas de classificação de *streaming* de dados. Para classificação, duas abordagens são sugeridas: uma chamada de classificador de Correção Temporal (*Temporal Correction classifier*) em que o modelo preditivo é adaptado para suportar o conceito, e outra chamada de classificador Temporalmente Aumentado (*Temporally Augmented classifier*), que propõe a utilização do processo de *feature engineering* com a vantagem de não requerer modificações na estrutura do classificador permitindo, assim, a utilização de qualquer algoritmo sem a necessidade de alteração de seu código. Neste trabalho, o conceito de **classificador Temporalmente Aumentado** é adaptado para problemas de regressão de séries univariadas. A ideia consiste em criar atributos que

representem a dependência entre as características de entrada e as observações anteriores de cada série, utilizando-os como entrada para o processo de aprendizagem dos algoritmos DSM.

A Figura 16 apresenta a abordagem usada neste estudo para criar atributos com base nos valores dos coeficientes ACF. O diagrama apresenta a criação de características para uma série temporal específica e o algoritmo apresentado na Figura 17 explica o processo em mais detalhes.

Figura 16 – Diagrama de criação de atributos derivados de coeficientes ACF.



Fonte: Autoria própria

Os dados de treinamento da série são lidos e analisados de forma iterativa, uma observação da série de cada vez, até que toda a série tenha sido processada. Para cada

Figura 17 – Algoritmo do processo de *feature engineering* (criação de atributos derivados de coeficientes ACF).

Algoritmo – *Feature Engineering*

```
1:  função featureEngineering(série temporal):
2:      S ← série temporal
        // percorre todas as observações da série S
        // para computar seus novos atributos
3:      para N = 1 até comprimento(S) faça:
            // dados da janela deslizante (tipo de dado: série temporal):
            // seleciona até 288 eventos anteriores ao registro atual N na série S
4:          W288 ← S[N-288-1 .. N-1]
            // calcula os coeficientes ACF para os dados da janela deslizante
5:          ACF_W288 ← forecast::Acf(W288, plot=FALSE)
            // normaliza os primeiros 18 coeficientes ACF
            // para a faixa de 1 a 2 para evitar valores zerados ou negativos
6:          ACF_W288_norm ← BBmisc::normalize(ACF_W288$acf[1:18],
                                    method="range", range = c(1,2))
            // seleciona os últimos 18 registro da janela deslizante em ordem inversa
7:          W288_18r ← reverse(tail(W288,18))
            // multiplica os valores normalizados de ACF pelos últimos 18 valores da
            // janela deslizante para computar os 18 novos atributos baseados nos
            // coeficientes ACF
8:          ACF_Lag ← ACF_W288_norm * W288_18r
            // cria atributo adicional baseado num forecast (one-step-ahead) usando
            // algoritmo de regressão linear que considera tendência e sazonalidade
9:          fit ← forecast::tslm(W288 ~ trend + season)
10:         fcTSLM_h1 ← forecast(fit, h=1)$mean
            // novos atributos ACF_Lag1 a ACF_Lag18, e fcTSLM_h1
            // estão prontos para serem agregados aos dados da série original
11:         S[N] ← bind(S[N], ACF_Lag1..ACF_Lag18, fcTSLM_h1)
12:     fim para
        // grava série S e seus novos atributos em arquivos externo ARFF
13:     grava série S em arquivo ARFF
14: fim função
```

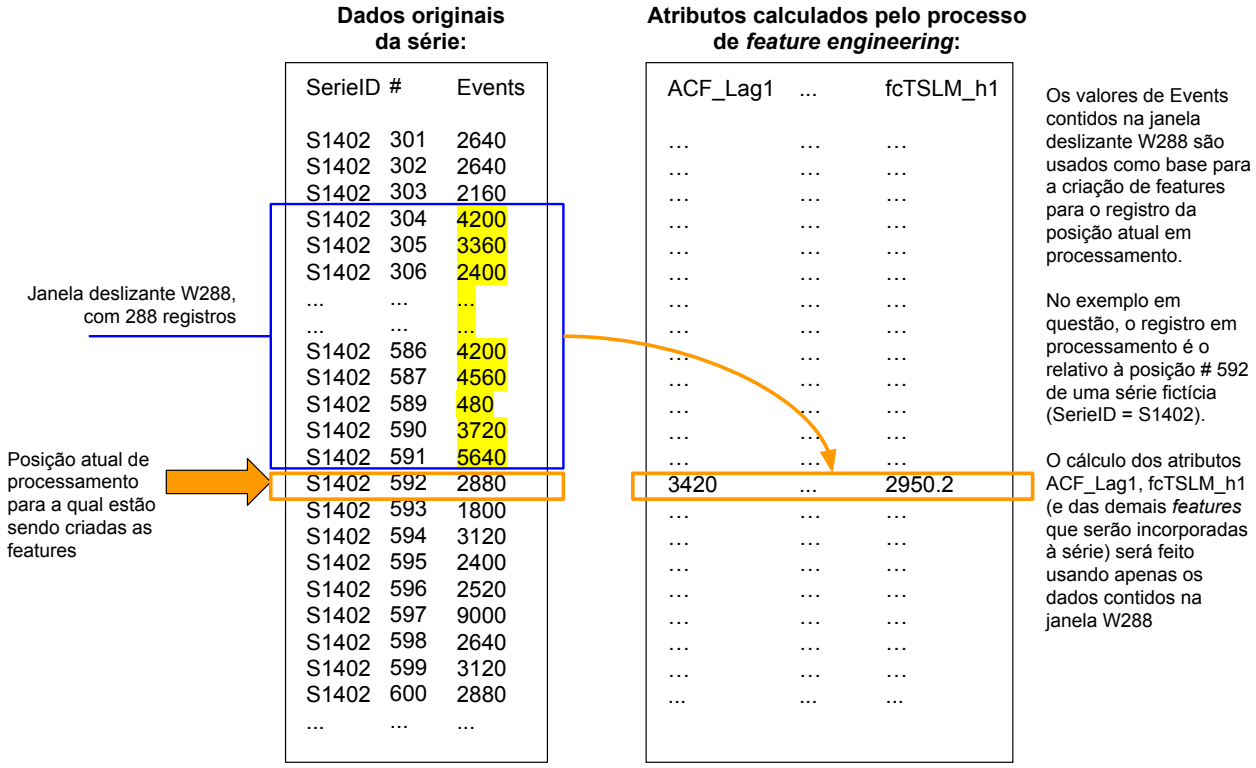
Fonte: Autoria própria

registro uma janela deslizante contendo as observações da série é criada. Com isso, uma janela pode conter de zero a 288 registros (W288), dependendo do número de ordem do registro que está sendo processado e, também, do comprimento original da série (pois, para séries com menos de 288 registros, nunca será possível obter uma janela com 288 observações).

É importante observar que os registros considerados nessa janela são registros anteriores ao que está sendo processado. Isso é feito de forma a evitar *data leakage* que poderia ocorrer pela incorporação no modelo de aprendizagem de informações que, na

etapa de predição, não estariam disponíveis. A Figura 18 mostra um exemplo de uso da janela deslizante para a criação de características (*features*) baseada numa janela de 288 registros.

Figura 18 – Exemplo de uso de janela deslizante para a seleção de dados a serem considerados na criação de características.



Fonte: Autoria própria

Uma vez obtidos os dados da janela W288 (que pode conter de 0 a 288 registros), os coeficientes ACF são calculados considerando os dados que ela contém, que basicamente representam uma série temporal parcial com até 288 observações.

Os primeiros 18 coeficientes ACF calculados com base na série de dados de W288 são normalizados para o intervalo de valores de 1 a 2 (a fim de evitar valores negativos ou zerados) e, em seguida, multiplicados pelos últimos 18 eventos da janela deslizante (os mais recentes), resultando nas características ACF (ou coeficientes de autocorrelação ajustados) que são adicionados à lista de atributos do registro que está sendo processado. Além disso, um atributo extra é criado com base em um modelo de regressão linear considerando os eventos da janela W288. Esse último atributo foi incorporado ao modelo para auxiliar no processo de treinamento com algoritmos DSM especialmente porque, para séries curtas ou enquanto não existirem registros suficientes para preencher uma janela deslizante, um atributo adicional referente a uma predição criada por um método de regressão pode ajudar

a incorporar uma *feature* que ajude a manter o modelo de aprendizagem do algoritmo DSM ligeiramente calibrado, pela incorporação de um atributo que apresenta um valor próximo à faixa de valores reais da série.

O processo de *feature engineering* cria um arquivo ARFF *(Attribute-Relation File Format)* para a série com a estrutura apresentada na Tabela 5.

Tabela 5 – Lista de atributos do arquivo ARFF que será usado no processo de aprendizagem com algoritmo DSM

Atributo	**Descrição**
SerieID (caracter)	Código identificador da série
NumOrdem (numérico)	Número de ordem (sequencial) da observação
Events (classe, numérico)	Valor da observação da série
ACF_Lag1 ... ACF_Lag18 (numérico)	Atributos calculados com base nos coeficientes de autocorrelação (ACF) para janela deslizante de 288 observações, normalizados para intervalo de 1 a 2 (a fim de evitar valores negativos ou zerados), multiplicados pelos últimos 18 valores das observações da série disponíveis na janela.
fcTSLM_h1 (numérico)	*Forecast one-step-ahead* calculado por um modelo de regressão para uma janela deslizante de 288 observações

Fonte: Autoria própria

Em resumo, o processo de *feature engineering* adiciona um indicador sequencial das observações da série, 18 atributos calculados com base em coeficientes de autocorrelação e um valor predito com base em um algoritmo de regressão linear. A opção pela escolha de janelas deslizantes de 288 observações e 18 coeficientes de autocorrelação foi feita a partir de experimentos descritos no Apêndice B deste estudo, que avaliou o desempenho de janelas de 36 a 480 eventos e o uso de 12, 18, 24, 36, 48 e 60 coeficientes ACF. Os valores finais foram selecionados com base na combinação que apresentou melhores resultados gerais nos experimentos realizados.

3.2.3 Atividades da Etapa de TREINAMENTO

Com os dados das séries temporais carregados e preparados com a criação de atributos adicionais para a série, passa-se para a etapa de treinamento (*fitting*), cujas atividades são descritas a seguir.

Atividades #2.1 - Treinamento (auto.arima) e #2.2 Cálculo de Erro de Treinamento (auto.arima)

Para o treinamento com o algoritmo estatístico ARIMA (Atividade #2.1), o método utiliza a função *forecast::auto.arima*, disponível para a linguagem R, que identifica de forma automática os melhores parâmetros para a execução desse algoritmo de regressão.

O treinamento com auto.arima é feito com base nos dados de treinamento da série, sem necessidade de transformação nos dados originais. O uso de auto.arima facilita o processo de treinamento pois nenhum tipo de ajuste de parâmetros é necessário.

A instrução a seguir (em Linguagem R) é executada:

```
fitmodel <- auto.arima(seriesToForecast)
```

onde *seriesToForecast* contém os dados da série.

Os n registros finais dos dados de *fitting* (treinamento) gerados pelo modelo e armazenados na variável *fitmodel$fitted* são de interesse principal para o processo de treinamento. Nesse caso, n corresponde ao total de registros que serão usados para o cálculo do erro de treinamento e é igual ao horizonte de predição h (quantidade de *forecasts*) estabelecidos de acordo com a periodicidade da série.

Após o treinamento com auto.arima, é feito o cálculo de erro de treinamento (Atividade #2.2). Para esse cálculo, os dados de treinamento gerados pelo modelo selecionado por auto.arima são confrontados com os dados originais da série e o erro é obtido. A porção dos dados considerados no cálculo do sMAPE de treinamento são os n últimos registros finais dos dados gerados pelo processo de *fitting* e dados reais da série.

O cálculo de erro de treinamento é feito com o uso da função `Metrics::smape()` do R, conforme abaixo:

```
erroTreinoNregsAutoArima <- smape(tail(seriesToForecast,numForecasts),
                                  tail(fitmodel$fitted,numForecasts))
```

onde *numForecasts* corresponde ao horizonte de predição h e coincide com o total de registros n a serem considerados no cálculo de erro de treinamento.

O sMAPE de treinamento calculado nessa etapa será usado posteriormente na atividade de seleção de algoritmo (Atividade #4.1).

Atividades #3.1 - *Prequential* (treinamento)(AdaGrad) e #3.2 - Cálculo de erro de Treinamento (AdaGrad)

Na Atividade #3.1 o objetivo principal é realizar o treinamento com o algoritmo AdaGrad. A atividade é denominada de *prequential* por utilizar o modelo de aprendizagem *Test-then-Train* (Testa-e-então-Treina) no qual o modelo de aprendizagem é desenvolvido de forma gradual à medida que cada registro do arquivo de dados é avaliado. Nesse modelo, uma predição é realizada e confrontada com o valor esperado e, em seguida, o modelo é "treinado" ou ajustado de forma a melhor se adaptar à variação dos dados.

O processo é gerenciado por um programa escrito em linguagem R que carrega o

arquivo ARFF com *features* ACF gerado na Atividade #1.2 *Feature Engineering* e o envia para treinamento com AdaGrad no *software* MOA.

A instrução a seguir (*moa.DoTask EvaluatePrequentialRegression*) corresponde à chamada externa que é feita ao MOA a partir do programa em R que gerencia o processo:

```
javaOutput <- system(paste('java -cp moa.jar -javaagent:sizeofag-1.0.4.jar
        moa.DoTask "EvaluatePrequentialRegression -l (functions.AdaGrad
        -r 0.01) -s (ArffFileStream -f (',folderARFFforecasts_AB2,'SF_',
        mmSerieID,'_FIT_AtributosDaData_TESTE_AB2_',
        mmNumMesForecast,'R.arff) -c 1) -e
        BasicRegressionPerformanceEvaluator -f 200000 -q 200000 -o (',
        folderOutputARFFforecasts_AB2,'OUTPUT_SF_',mmSerieID,
        '_FIT_AtributosDaData_TESTE_AB2_',mmNumMesForecast,'R.csv)"',
        sep=), intern = FALSE)
```

Nessa intrução são utilizados os seguintes parâmetros principais de configuração:

- *task*: *EvaluatePrequentialRegression*
- *learner (-l)*: AdaGrad *(functions.AdaGrad)* com parâmetros *default (learningRate=0,01; epsilon=0; lambdaRegularization=0; lossFunction=HINGE).*
- *stream (-s)*: nome do arquivo ARFF (-f) que será usado no processamento e valor a ser considerado para a *classIndex (-c)* que indica qual o atributo a ser considerado como *target* da predição.
- *evaluator (-e)*: *BasicRegressionPerformanceEvaluator*
- *sampleFrequency (-f)*: 200.000
- *memCheckFrequency (-q)*: 200.000
- *outputPredictionFile (-o)*: nome do arquivo de *output* gerado pelo MOA com dados do processo de teste/treinamento/*prequential.*

Com essa instrução é feita uma chamada ao MOA, que executa o algoritmo AdaGrad usando seus parâmetros *default* (*learning rate* = 0.01) no processamento do arquivo ARFF gerado no processo de *feature engineering.* A execução no MOA gera um arquivo de saída no formato texto que contém os dados de *fitting* (treinamento) gerados por AdaGrad e os dados originais da série.

O cálculo de erro de treinamento (Atividade #3.2) é feito com a função `Metrics::smape()` do R, conforme abaixo :

```
erroTreinoNregsAdaGrad <- smape(tail(arqOriginal$Events,numForecasts),
                                tail(vectorFITTINGValues, numForecasts))
```

onde *arqOriginal$Events* contém os dados originais da série; *numForecasts* corresponde ao horizonte de predição h e que coincide com o total de registros n a serem considerados no cálculo de erro de treinamento; *vectorFITTINGValues* contém os valores extraídos do arquivo de saída gerado pelo MOA com os dados de treinamento (*fitting*) da série.

O erro de treinamento calculado nesta etapa será usado posteriormente como base no processo de seleção de algoritmo (Atividade #4.1).

3.2.4 Atividades da Etapa de SELEÇÃO

Nessa seção são descritas as atividades relativas ao processo de seleção de algoritmo que determina qual o melhor algoritmo a ser usado para a predição de dados da série em análise.

Atividade #4.1 Seleção de Algoritmo

O processo de Seleção de Algoritmo utilizado pelo método desenvolvido é implementado como uma comparação simples entre os valores de sMAPE de treinamento obtidos por auto.arima (Atividade #2.2) e AdaGrad (Atividade #3.2).

Ele é responsável pela identificação do algoritmo que apresentou menor valor de sMAPE na etapa de treinamento. Se o erro de treinamento de AdaGrad for menor que o erro de treinamento apesentado por auto.arima, então AdaGrad será o algoritmo selecionado, senão auto.arima será selecionado para a definição dos *forecasts* da série.

3.2.5 Atividades da Etapa de *FORECASTING*

Uma vez estabelecido o algoritmo recomendado para a série na Atividade #4.1, passa-se para a etapa de *FORECASTING*, em que são realizadas as predições da série e que correspondem ao resultado principal do método.

Atividade #5.1 - *Forecasting* (auto.arima)

A predição com auto.arima é feita com base no modelo de aprendizagem (variável *fitmodel*) gerado na etapa de treinamento (Atividade #2.1 Treinamento (auto.arima)):

```
fitmodel <- auto.arima(seriesToForecast)
```

onde *seriesToForecast* contém os dados da série.

Uma vez que o modelo já está estabelecido, os *forecasts* de auto.arima são gerados com a seguinte instrução:

```
forecastsAutoArima <- forecast(fitmodel, h=numForecasts)$mean
```

onde *numForecasts* corresponde ao total de *forecasts* (horizonte de predição, h) estabelecidos para a série.

Caso auto.arima tenha sido selecionado pelo processo de seleção (Atividade #4.1 Seleção de Algoritmo) como o algoritmo ideal para a série, essas predições representam os *forecasts* finais do método para a série, ou seja:

```
forecastsMetodo <- forecastsAutoArima
```

Atividade #5.2 - Prequential/Forecasting (AdaGrad)

Nessa atividade, o objetivo principal é obter as predições da série com o algoritmo de mineração de fluxos de dados AdaGrad usando o arquivo com dados de treinamento preparado na Atividade #1.2 que inclui, além das características básicas da série, os atributos gerados pelo processo de *Feature Engineering*.

A atividade é denominada de *prequential* por utilizar o modelo de aprendizagem *Test-then-Train* (Testa-e-Então-Treina) no qual o modelo de aprendizagem é desenvolvido de forma gradual à medida que cada registro do arquivo de dados é avaliado. Nesse modelo, uma predição é realizada e confrontada com o valor esperado e, em seguida, o modelo é "treinado", ajustado de forma a melhor se adaptar à variação dos dados.

Para o processo de predição é utilizado um programa escrito em linguagem R que carrega o arquivo ARFF com dados da série (dados originais e atributos adicionais) e que dispara o processo de mineração de fluxos de dados no MOA a fim de realizar o cálculo dos *forecasts* de testes da série.

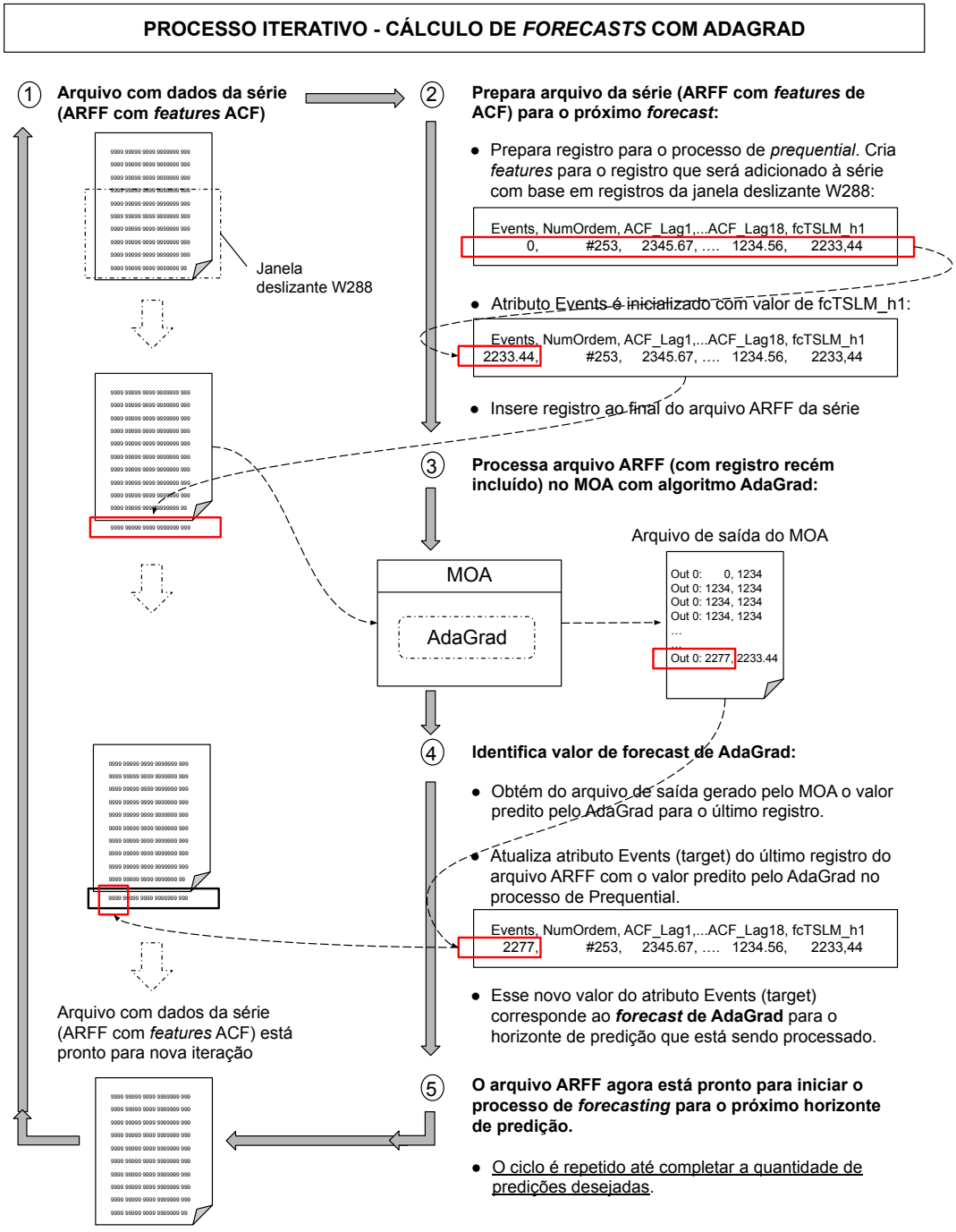
As predições são realizadas no modelo *one-step-ahead forecast* e de forma iterativa, em que o *forecast* gerado para um determinado horizonte é reinserido na série para o predição de valor da série para o próximo horizonte.

A Figura 19 explica como se dá o processo iterativo de geração de *forecasts* com AdaGrad. O processo de *feature engineering* e os passos necessários para preparar o arquivo ARFF para cada predição são gerenciados por um programa em R e a predição com o AdaGrad é realizada no *software* MOA (usando a instrução (*moa.DoTask EvaluatePrequentialRegression*) descrita na Atividade #3.1).

O processo de criação de características utiliza os mesmos procedimentos que os descritos na Atividade #1.2, com a diferença que, nesta fase, são criados apenas os atributos necessários para criação do próximo registro a usar para a predição.

A cada iteração do processo de predição para um novo horizonte h, um novo arquivo ARFF é criado contendo os dados do arquivo processado anteriormente acrescidos de um novo registro cujos atributos foram recém-criados pelo processo de *feature engineering*.

Figura 19 – Processo iterativo de geração de forecasts com AdaGrad



Fonte: Autoria própria

Uma vez que os *forecasts* estão gerados, os arquivos ARFF criados pelo processo podem ser excluídos.

O ciclo de predição é repetido de acordo com a quantidade de *forecasts* desejados, definidos pela periodicidade da série no momento da inicialização de parâmetros do método.

Uma variável denominada *forecastsAdaGrad* armazena as predições feitas por esta Atividade #5.2. No caso de o método estar configurado para não fazer uso de fusão de *forecasts*, essas predições representarão os *forecasts* finais gerados pelo método. Caso contrário, essas predições serão usadas no processo de fusão com *forecasts* de auto.arima descrito nas atividades #5.3 e #5.4.

Atividade #5.3 - Forecasting (auto.arima)

Esta atividade é executada nos casos em que o método está configurado para fazer uso do processo de fusão. Sua descrição corresponde à apresentada para a atividade #5.1 - *Forecasting* (auto.arima) com a diferença que as predições geradas nesta atividade não serão os *forecasts* finais do método, mas passarão por processo de fusão com os *forecasts* de AdaGrad (Atividade #5.4).

Atividade #5.4 - Fusão de *forecasts*

Esta Atividade #5.4 - Fusão de *forecasts* será executada nos casos em que o algoritmo selecionado para a série seja o AdaGrad e que o método esteja configurado para realizar a fusão (configuração *default*).

Consiste basicamente no cálculo de uma média dos valores de *forecasts* de auto.arima (gerados pela Atividade #5.3) com os valores de AdaGrad (gerados pela Atividade #5.2). Experimentos prévios descritos no Apêndice B (Experimento 3.7) mostram que o uso dessa abordagem é capaz de apresentar, para determinados cenários, resultados melhores que o uso isolado dos valores de AdaGrad.

Com isso, os *forecasts* finais do método para a série serão obtidos com a seguinte expressão:

```
forecastsMetodo <- mean(forecastsAutoArima, forecastsAdaGrad)
```

3.2.6 Atividades da Etapa de TESTES

A etapa de TESTES está incluída no diagrama do método com o objetivo de sugerir um conjunto completo de atividades para o processo de predição, incluindo desde as atividades de preparação de dados, passando pelos processos de treinamento, seleção, forecasting/fusão e testes. Inclui, essencialmente, a atividade #6.1 que tem a função de verificar a acurácia do método de predição das séries submetidas ao método.

Atividade #6.1 - Cálculo de sMAPE de teste

Para o cálculo do erro de teste, tomam-se como base as predições geradas na etapa de *FORECASTING* do método e os valores de testes disponibilizados para a série.

O erro é calculado para cada horizonte e um sMAPE médio é obtido para o total de predições efetuadas para a série de forma a estabelecer a acurácia do método para a série.

Para o cálculo de erro para um determinado horizonte de predição (h1, h2,...), usa-se a expressão:

```
sMAPE para horizonte h <- smape(valor original da série para posição h,
                                valor previsto pelo método para a posição h)
```

Para o cálculo do sMAPE de teste para a série (todos os horizontes), pode-se usar a expressão:

```
sMAPE da série <- smape(dadosDeTeste, forecastsMetodo)
```

3.3 Demonstração do Método

Nesta seção, é feita uma demonstração do método explicando passo a passo os processos envolvidos na preparação e na geração de *forecasts* para uma série-exemplo. Algumas expressões apresentadas na seção 3.2 - Descrição do Método são reapresentadas aqui com o objetivo de tornar a demonstração mais clara.

A demonstração é realizada com o método em sua configuração *default* (ou método Seleção e Fusão), que faz uso do processo de fusão de *forecasts* nos casos em que o algoritmo AdaGrad tenha sido selecionado para a série em análise.

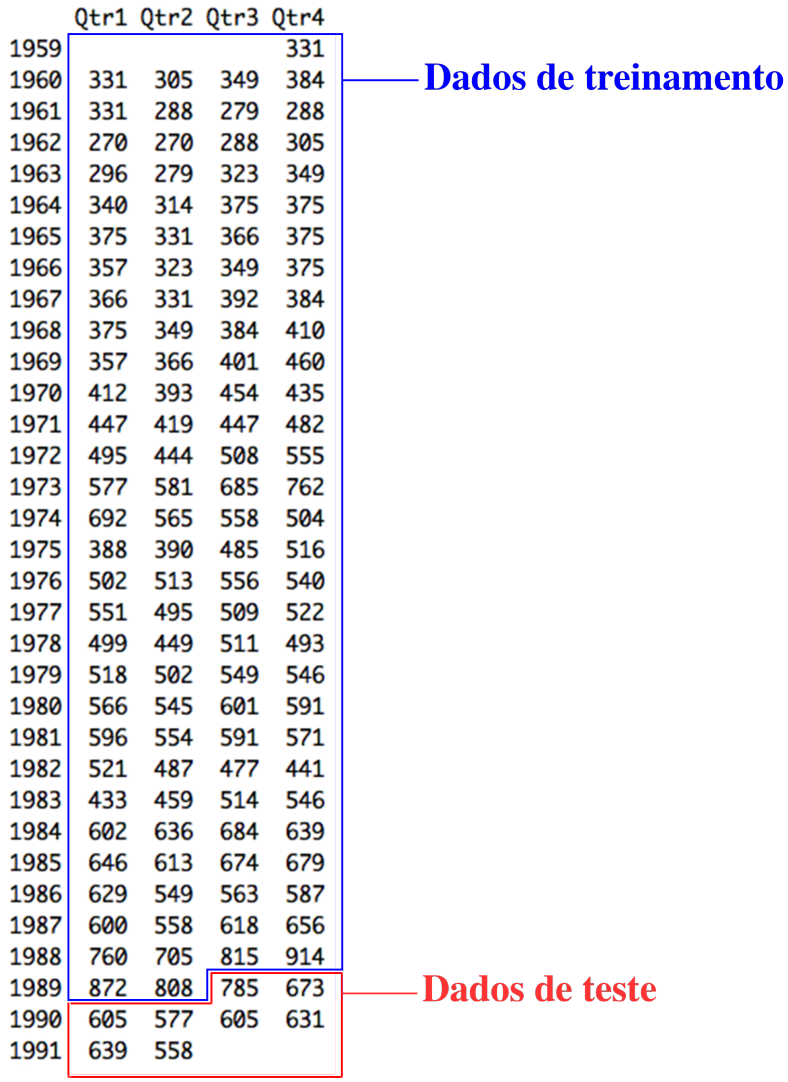
Carregamento de Dados da Série

A Figura 20 apresenta os dados originais da série 16, de frequência 4, periodicidade trimestral (*quarterly*), do *dataset* TSDL (descrito em mais detalhes na seção Experimentos) e destaca a forma como estão divididos em dados de treinamento e de teste. Essa série será utilizada na demonstração descrita nesta seção e será denominada deste ponto em diante como série-exemplo.

A Figura 21 apresenta um gráfico com os dados de treinamento da série-exemplo. A série original possui 127 observações; no entanto, para a demonstração, serão utilizadas suas 119 observações iniciais para treinamento e suas 8 finais serão reservadas para dados de teste.

O método faz a predição de cada série de forma individual. Para o processamento de *datasets* com várias séries temporais, o programa escrito em R que gerencia a sequência de passos do método possui uma seção inicial em que são feitas configurações quanto à forma de carregamento e identificação dos dados de treinamento e de testes das séries.

Figura 20 – Dados originais de uma série-exemplo, extraída do dataset TSDL, frequência 4, série 16. Estão indicados os dados de treinamento e dados de teste da série.



Fonte: Autoria própria

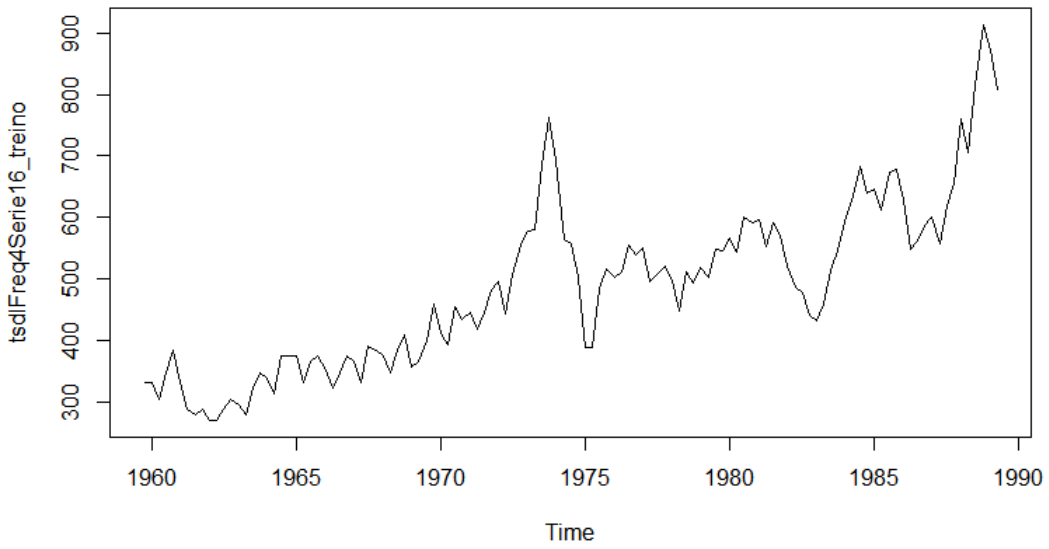
Uma vez carregados os dados originais da série, os demais passos do método são realizados, conforme exemplificados nos passos descritos a seguir.

Feature Engineering

O processo de *feature engineering* é realizado tomando como base os dados de treinamento da série.

O arquivo ARFF gerado para a série nesse processo possui o formato apresentado na Figura 22, que mostra apenas um trecho do documento a título de ilustração do aspecto geral do arquivo.

Figura 21 – Série-exemplo, extraída do dataset TSDL, frequência 4, série 16. O gráfico representa os dados de treinamento da série, que ignoram os 8 registros finais da série original.



Fonte: Autoria própria

É possível observar que, para os primeiros registros do arquivo ARFF gerado, vários campos estão zerados e, gradualmente, vão recebendo valores diferentes de zero. Isso ocorre devido ao uso de janelas deslizantes na geração dos atributos. Então, à medida que as janelas de dados vão recebendo maior quantidade de observações da série, os coeficientes de autocorrelação ACF são gerados de forma a completar os atributos de cada registro.

Treinamento com auto.arima e Cálculo de Erro de Treinamento

O treinamento com auto.arima é feito com base nos dados de treinamento da série. A instrução a seguir (em Linguagem R) é executada:

```
fitmodel <- auto.arima(seriesToForecast)
```

onde *seriesToForecast* contém os dados da série.

Após a execução dessa instrução, a variável *fitmodel* contém informações geradas com o treinamento com o método auto.arima, sendo de interesse principal o conteúdo de *fitmodel$fitted*, que apresenta os seguintes valores (apresentando apenas os 10 valores iniciais e 10 valores da série; os últimos 8 valores serão usados no cálculo do erro de treinamento):

```
fitmodel$fitted = 331, 331, 305, 349, 384, 331, 288, 279, 288, 270,...,
```

Figura 22 – Estrutura do arquivo ARFF gerado pelo processo de *Feature Engineering* para uma série-exemplo.

Fonte: Autoria própria

```
587, 600, 558, 618, 656, 760, 705, 815, 914, 872
```

Considerando que o método estabelece que o erro de treinamento é calculado com base em n registros finais de treinamento (onde n corresponde ao horizonte de predições h estabelecido de acordo com a frequência da série), o erro é calculado considerando os últimos 8 registros.

O cálculo de erro de treinamento é feito com o uso da função `Metrics::smape()` do R, conforme abaixo:

```
erroTreinoNregsAutoArima <- smape(tail(seriesToForecast,numForecasts),
                                  tail(fitmodel$fitted,numForecasts))
```

onde *numForecasts* corresponde ao horizonte de predição h e que coincide com o total de registros a serem considerados no cálculo de erro de treinamento.

Substituindo no cálculo de SMAPE as variáveis pelos valores atuais da série e os

valores de treinamento, tem-se o equivalente a:

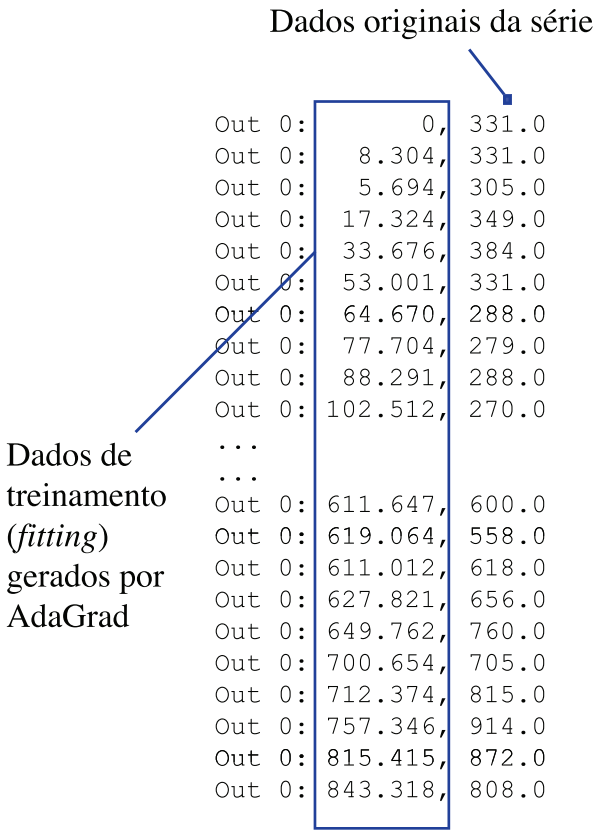
```
erroTreinoNregsAdaGrad <- smape(c(618, 656, 760, 705, 815, 914, 872, 808),
                                c(558, 618, 656, 760, 705, 815, 914, 872))
```

O erro de treinamento calculado por auto.arima para a série-exemplo é:
`erroTreinoNregsAutoArima = 0.096`

Treinamento com AdaGrad e Cálculo de Erro de Treinamento

O treinamento com AdaGrad é feito com base nos dados de treinamento da série, armazenados no arquivo ARFF que foi gerado pelo processo de *feature engineering*. A execução de AdaGrad no MOA gera um arquivo com os dados de treinamento. Na Figura 23, é apresentado um trecho dos valores gerados para a série-exemplo, com destaque para os dados gerados por AdaGrad. Os 8 valores finais desses dados gerados por AdaGrad serão utilizados no cálculo de erro de treinamento.

Figura 23 – Trecho do arquivo de saída gerado pela execução de AdaGrad no MOA. Dados do processamento de uma série-exemplo.



Fonte: Autoria própria

O cálculo de erro de treinamento é feito com a função `Metrics::smape()` do R, conforme abaixo :

```
erroTreinoNregsAdaGrad <- smape(tail(arqOriginal$Events,numForecasts),
                          tail(vectorFITTINGValues, numForecasts))
```

onde *arqOriginal$Events* contém os dados originais da série; *numForecasts* corresponde ao horizonte de predição h , que coincide com o total de registros n a serem considerados no cálculo de erro de treinamento; *vectorFITTINGValues* contém os valores extraídos do arquivo de saída gerado pelo MOA com os dados de treinamento (*fitting*) da série.

Substituindo no cálculo de sMAPE as variáveis pelos valores atuais da série e os valores de treinamento, tem-se o equivalente a:

```
erroTreinoNregsAdaGrad <- smape (c(618, 656, 760, 705, 815, 914, 872, 808),
c(611.012, 627.821, 649.762, 700.654, 712.374, 757.346, 815.415, 843.318))
```

O erro de treinamento calculado para a série-exemplo em relação ao treinamento com AdaGrad é:

```
erroTreinoNregsAdaGrad = 0.081
```

Seleção de Algoritmo

A seleção de algoritmo implementada no método é muito simples: o algoritmo que apresentar menor erro de treinamento será considerado como o método mais adequado para a série.

Comparando os valores de *erroTreinoNregsAutoArima* (0,096) e *erroTreinoNregsAdaGrad* (0,081), essa etapa seleciona AdaGrad como o método a ser considerado para a série, pois *erroTreinoNregsAdaGrad* é menor que *erroTreinoNregsAutoArima*.

Forecasts com auto.arima

A predição com auto.arima é feita com base no modelo de aprendizagem gerado na etapa de treinamento (variável *fitmodel*).

```
fitmodel <- auto.arima(seriesToForecast)
```

onde *seriesToForecast* contém os dados da série.

Uma vez que o modelo já está estabelecido, os *forecasts* de auto.arima são gerados com a seguinte instrução:

```
forecastsAutoArima <- forecast(fitmodel, h=numForecasts)$mean
```

onde *numForecasts* corresponde ao total de *forecasts* (horizonte de predição) estabelecidos para a série, no caso h=*8*.

Para a série-exemplo os *forecasts* de auto.arima estão indicados a seguir. No caso, os valores médios (forecast()$mean) estabelecidos por auto.arima foram todos iguais sugerindo que o método prevê uma estabilidade por todo o horizonte de predição:
`forecastsAutoArima: 808, 808, 808, 808, 808, 808, 808, 808`

Forecasts com AdaGrad

Para a predição com AdaGrad é utilizada a mesma abordagem geral que a utilizada para o treinamento. Ou seja, o arquivo ARFF criado com os atributos gerados com base em coeficientes de autocorrelação ACF é submetido para treinamento no MOA. Porém, o processo é repetido h vezes (onde h = *horizonte de predição*) de forma a calcular um *forecast* de AdaGrad a cada passo, até se obter o total de predições estabelecidas de acordo com a periodicidade da série.

O procedimento é otimizado para não precisar ser repetido em sua totalidade. Considerando que o arquivo ARFF original com dados de toda a série já tinha sido gerado na fase de *feature engineering* e que a criação do modelo inicial do AdaGrad com a série completa (todos os dados de treinamento) já foi realizado no MOA na etapa de treinamento, o que ocorre nessa fase é a adição gradual de registros no arquivo ARFF, um para cada predição a ser realizada. Cada arquivo gerado é reprocessado no MOA e o arquivo de saída gerado no processo é usado como base para realimentar o processo. Para isso, é extraído desse arquivo o valor predito pelo MOA para o próximo horizonte e ele é inserido num novo arquivo ARFF junto com os registros anteriores da série. Em seguida, seus atributos adicionais (ACF) são calculados e inseridos no ARFF que será usado na próxima execução do AdaGrad.

Esse modelo iterativo corresponde a dizer que serão feitas h predições *one-step-ahead* para AdaGrad até se obter o total de *forecasts* (h) necessários de acordo com a periodicidade da série (para a série-exemplo, esse total é $h = 8$ *forecasts*).

Para a série-exemplo, os valores originais preditos por AdaGrad são os seguintes:
`forecastsAdaGrad: 841.301, 856.905, 869.844, 880.819, 903.508, 937.82, 975.376, 1011.602`

Forecasts Finais do Método

Os *forecasts* finais do método para uma determinada série dependem do algoritmo selecionado como aquele que apresentou menor erro no processo de treinamento.

No caso de auto.arima apresentar o menor erro de treinamento, seus *forecasts* serão os valores preditos para a série.

Por sua vez, caso AdaGrad tenha apresentado menor erro de treinamento (que é o caso para a série-exemplo em análise), seus *forecasts* podem passar por um processo

de fusão com os de auto.arima de forma a estabelecer os valores finais da série. Como explicado anteriormente, esta demonstração está sendo realizada utilizando o método em sua configuração *default*, que faz uso do processo de fusão.

A fusão proposta para o método corresponde à aplicação de uma média simples entre os *forecasts* calculados pelos algoritmos para cada horizonte intermediário (*h1 a h8*). A Figura 24 mostra o processo de fusão aplicado à série de exemplo:

Figura 24 – Demonstração do processo de fusão de forecasts para uma série-exemplo.

	Horizonte de predição							
	h1	h2	h3	h4	h5	h6	h7	h8
Forecasts de auto.arima	808	808	808	808	808	808	808	808
Forecasts de AdaGrad	841,301	856,905	869,844	880,819	903,508	937,82	975,376	1011,602
Fusão (Média)	824,651	832,453	838,922	844,410	855,754	872,910	891,688	909,801

Valores calculados pela fusão (média) de *forecasts* de auto.arima e AdaGrad

Fonte: Autoria própria

Com isso, os valores finais de *forecasts* calculados para a série-exemplo são os seguintes:

```
forecasts do método = 824.651, 832.453, 838.922, 844.410, 855.754,
                      872.910, 891.688, 909.801
```

Cálculo de Erro de Teste

Uma vez calculados os *forecasts* do método para a série em análise, pode ser feita uma verificação do erro de teste gerado pelo processo (caso os valores de teste sejam conhecidos para a série).

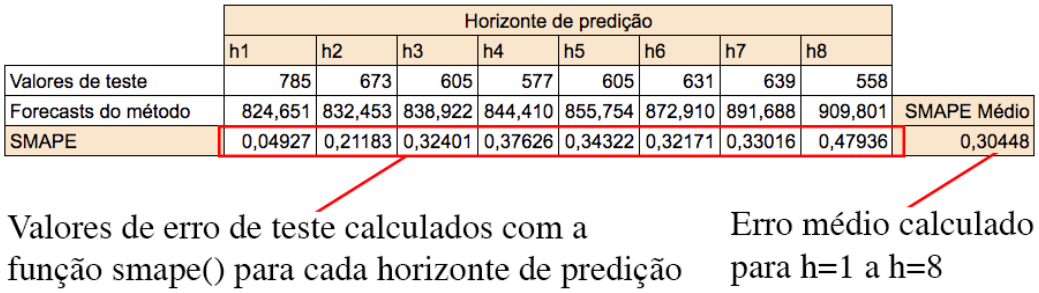
Considerando que os valores de teste da série-exemplo são os 8 registros finais reservados da série no início do processo e tomando como base os valores de *forecasts* determinados pelo método (que no caso é resultado da fusão de *forecasts* de auto.arima e AdaGrad), o erro pode ser calculado para cada horizonte conforme expressão abaixo:

```
sMAPE para horizonte h = smape(valor original da série para posição h,
                               valor previsto pelo método para a posição h).
```

A Figura 25 mostra os valores de teste da série, os respectivos valores previstos para cada horizonte e o valor de sMAPE para cada posição.

Os valores de erro (sMAPE) são calculados para cada horizonte. O erro médio para o horizonte completo de predição (no caso 8 trimestres) corresponde à média dos

Figura 25 – Demonstração do processo de cálculo de SMAPE de teste para uma série-exemplo.



Fonte: Autoria própria

sMAPEs calculados para cada horizonte, que estabelece o sMAPE para a série para todo o horizonte de predição.

No caso, o sMAPE de teste calculado para a série-exemplo é igual a :

```
SMAPE de teste para h1 a h18 = 0.30448, que equivale a 30,448%
```

Os passos apresentados nessa demonstração correspondem ao processo de predição de uma série temporal. Para a predição das séries de um mesmo *dataset*, o processo é repetido para cada série individualmente.

3.4 Técnicas Alternativas

No método proposto, o erro de treinamento calculado (utilizado posteriormente no processo de seleção de algoritmo) leva em consideração os últimos n registros dos dados de treinamento, onde n corresponde ao horizonte de predição da série (h). Além disso, o método estabelece (em sua configuração *default*) o uso de fusão de *forecasts* a ser utilizada nos casos em que AdaGrad seja o algoritmo mais indicado para a série.

Essa estrutura é a que apresentou maior ganho médio geral em relação ao predição efetuado com o uso isolado de auto.arima nos experimentos realizados e descritos neste estudo na seção Experimentos. No entanto, técnicas alternativas foram avaliadas e merecem ser consideradas em aprimoramentos futuros do método, especialmente porque o ganho demonstrado por outras técnicas sugere que a avaliação de características e de comportamento do método em tempo de execução pode trazer resultados melhores para a série.

As técnicas avaliadas estão descritas a seguir e devem ser consideradas como passíveis de serem utilizadas individualmente ou de forma combinada. As técnicas citadas levam em consideração apenas os algoritmos estabelecidos para o método (auto.arima e

AdaGrad).

Uso isolado de auto.arima

O método ARIMA, como descrito anteriormente, é um método estatístico clássico que apresenta resultados satisfatórios em vários cenários. A implementação auto.arima foi proposta como elemento deste estudo para ser utilizada com seus parâmetros definidos automaticamente, sem nenhum tipo de customização. No entanto, é um método que suporta a alteração de parâmetros de forma a que avalie uma quantidade maior de modelos para as séries, ou que considere ou não a análise de sazonalidade ou estacionariedade, além de outras personalizações. Com isso, é um método que pode apresentar resultados individuais melhores que outros métodos, especialmente se os parâmetros estabelecidos por auto.arima forem os ideais para a série.

Uso isolado de AdaGrad

O algoritmo AdaGrad tem uma estrutura de funcionamento bem distinta da apresentada por auto.arima, em especial por adequar o seu modelo de forma gradual, adaptativa, sem dependência de análise prévia de todos os dados da série tal como ocorre em métodos *batch*, ou não orientados a fluxos de dados. Geralmente, depende de uma grande quantidade de registros para adequar o seu modelo e é mais indicado para cenários em que os dados sejam de natureza multivariada.

Apesar disso, é possível observar pelos experimentos realizados, que existem cenários em que AdaGrad é capaz de mostrar resultados melhores que auto.arima, em especial em séries que apresentem muita intermitência nos dados, com os valores alternando entre picos numéricos que se alternam com valores zerados nas séries (ver resultados com *dataset* M5 na seção Experimentos).

SeleçãoNRegs - Seleção com base em *n* registros, porém sem uso de processo de Fusão

O método proposto neste estudo estabelece que, após efetuar o processo de seleção de algoritmo, se o resultado da etapa de seleção de algoritmo indicar AdaGrad como o melhor método para a série, uma fusão dos *forecasts* de auto.arima e AdaGrad deve ser realizada. No entanto, essa abordagem nem sempre é capaz de trazer os melhores resultados para as séries. Apesar de essa técnica apresentar maior ganho médio geral para os experimentos realizados (e descritos posteriormente neste trabalho), é possível verificar que a fusão pode ser dispensada em alguns casos.

Séries que apresentem muita intermitência nos dados, ou que tenham uma característica de crescimento bem determinada e suave (sem muitas oscilações nos dados) podem se

beneficiar do uso dessa técnica, sem necessidade de uso de fusão dos *forecasts* de AdaGrad com auto.arima (ver resultados com *datasets* M5 e COVID na seção Experimentos).

Seleção40p - Seleção com base em erro de treinamento calculado com 40% dos dados de treinamento

Em experimentos prévios à definição da estrutura final do método proposto por este estudo, diferentes abordagens de cálculo de erro de treinamento foram avaliadas, considerando diferentes quantidades de registros, além do uso ou não de separação de dados de treinamento e de validação.

O uso de 40% dos registros finais da série se destacou no cálculo de erro de treinamento, apresentando desempenho melhor que outros percentuais, como 5%, 10% ou até mesmo o erro calculado com base em todos os registros da série.

Os resultados obtidos com o cálculo do erro considerando os 40% finais dos dados de treinamento podem ser, em muitos casos, superiores aos obtidos com o método proposto. No entanto, pensando em uma implementação orientada a fluxos de dados, separar os 40% dos registros finais de uma série, apesar de ser um procedimento fácil para processos em *batch*, não é necessariamente aplicável para uma estratégia em que não se tenha conhecimento prévio do total de registros que compõem a série. Por isso, seu uso deve ficar limitado aos casos em que as séries são de comprimento conhecido.

Proporção80/20 - Fusão de *forecasts* sem uso do processo de seleção

O uso de fusão de *forecasts*, em especial o uso da média é uma estratégia bem conhecida na combinação de diferentes métodos, (HYNDMAN; ATHANASOPOULOS, 2018).

Neste estudo, diferentes proporções foram avaliadas e foi observado que o uso de uma combinação de *forecasts* na proporção 80/20 (80% auto.arima, 20% AdaGrad) é capaz de apresentar resultados com ganhos superiores aos de auto.arima isolado ou das demais técnicas exploradas em diferentes *datasets*.

Se a técnica for utilizada em séries que não apresentem intermitência e que não tenham uma característica de crescimento suave tal como descritas para a SeleçãoNRegs, é possível obter bons resultados.

Sua principal vantagem está no fato de que dispensa a necessidade de cálculo de erro de treinamento e o processo de seleção de algoritmo. A inconveniência dessa técnica é que os *forecasts* devem ser, obrigatoriamente, calculados para ambos os algoritmos e não apenas com aquele que tenha sido escolhido pelo processo de seleção. Mas esse é um custo que parece valer a pena pela simplicidade que traz ao método.

3.5 Considerações Finais

Neste capítulo, foi apresentado um método de predição de multisséries temporais univariadas que implementa um *ensemble* com um método estatístico (auto.arima) e um algoritmo de mineração de fluxos de dados (AdaGrad).

O método proposto é resultado de um estudo exploratório que envolveu a análise de diferentes conceitos, abordagens, algoritmos e técnicas em busca de estabelecer um modelo competitivo com métodos do estado-da-arte no processo de predição. Os Apêndices A e B, detalham todos os experimentos realizados e as técnicas avaliadas de forma a identificar uma combinação capaz de superar a performance obtida com o uso isolado dos algoritmos envolvidos.

Na próxima seção (Experimentos e Resultados), são descritos os experimentos realizados com o método proposto neste capítulo. O objetivo dos experimentos é verificar o desempenho do método no processamento de diferentes *datasets*, com séries temporais de várias frequências e avaliar sua capacidade de gerar *forescasts* para diferentes horizontes de predição de forma a superar os resultados individuais obtidos pelo uso do método estatístico auto.arima.

4 Experimentos e Resultados

Este capítulo descreve os experimentos realizados com o método AA-ACF com o objetivo de avaliar o seu desempenho na predição de séries temporais de diferentes *datasets*.

Ao utilizar diferentes conjuntos de séries, o método é submetido a séries com características diversas quanto ao comprimento e aspecto dos dados das séries. Com isso, é possível verificar o comportamento do método em diferentes cenários de aplicação.

4.1 *Datasets*

Nesta seção são relacionados e descritos os *datasets* de séries temporais utilizados nos experimentos realizados com o método AA-ACF.

Para cada *dataset*, é apresentada uma breve descrição de sua origem ou composição e estatísticas gerais que apresentam informações quanto ao número de observações das séries (mínimo, máximo, média e desvio padrão). Essas estatísticas **são referentes aos dados reservados para o treinamento.**

4.1.1 *Dataset* M3

O *dataset* M3 utilizado neste experimento refere-se ao conjunto de séries temporais disponibilizadas para a competição M3 (MAKRIDAKIS; HIBON, 2000), que inclui um conjunto de 3.003 séries de diferentes domínios. Essa competição faz parte das *M-Competitions*, ou *Makridakis-Competitions* (https://mofc.unic.ac.cy/), cujos *datasets* são amplamente utilizados e discutidos no meio acadêmico, sendo possível encontrar vários artigos na literatura que fazem referência a tais competições.

As séries desse conjunto M3 representam informações extraídas de diferentes domínios (Microeconomia, Indústria, Macroeconomia, Finanças, Dados Demográficos e Outros).

Do ponto de vista de periodicidades e número de observações das séries, a Tabela 6 apresenta um resumo dos dados estatísticos de M3. Os valores de horizonte de predição (h) indicados na tabela se referem aos valores estabelecidos pela competição e expressam o número de *forecasts* a serem calculados para cada série. Os valores de frequência foram padronizados para todos os *datasets* envolvidos no experimento de acordo com a periodicidade ou tipo da série.

Para os experimentos, serão utilizadas as 3.003 séries do *dataset*.

Tabela 6 – *Dataset* M3 - Dados estatísticos das séries do *dataset*.

Dados Gerais das Séries				Número de Observações para Treinamento			
Periodicidade	Qtd	Frequência	h	Mín.	Máx	Média	Desvio Padrão
Monthly (M)	1428	12	18	48	126	99,34	28,50
Yearly (Y)	645	1	6	14	41	22,40	9,90
Quarterly (Q)	756	4	8	16	64	40,95	10,64
Other (O)	174	1	8	63	96	68,58	10,87
Total:	3003						

Fonte: Autoria própria

Observando os dados estatísticos das séries, pode-se notar que as séries são relativamente curtas, com comprimentos que variam de 14 a 126 eventos. Trata-se, portanto, de um conjunto que pode ajudar a verificar o comportamento do método em séries curtas.

4.1.2 *Dataset* M4

O *dataset* M4 utilizado neste experimento refere-se ao conjunto de séries temporais disponibilizadas para a competição M4, (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a), que inclui um total de 100.000 séries temporais.

A competição M4, além de conhecida por fazer parte do grupo de competições *M-Competitions*, é também uma competição amplamente divulgada pelo *International Institute of Forecasting* (https://forecasters.org/) em sua publicação oficial denominada *International Journal of Forecasting* (https://ijf.forecasters.org/). Ou seja, trata-se de um conjunto de séries selecionadas especificamente para uso por grupos de estudos focados no processo de predição de séries temporais, por isso, de grande interesse para este estudo.

Do ponto de vista de periodicidades e número de observações das séries, a Tabela 7 apresenta um resumo dos dados estatísticos de M4. Os valores de horizonte de predição (h) indicados na tabela se referem aos valores estabelecidos pela competição e expressam o número de *forecasts* a serem calculados para cada série. Os valores de frequência foram padronizados para todos os *datasets* envolvidos no experimento de acordo com a periodicidade ou tipo da série.

As séries desse conjunto M4 representam informações extraídas de diferentes domínios (Microeconomia, Indústria, Macroeconomia, Finanças, Dados Demográficos e Outros). Diferentemente do dataset M3 que contém séries de curta duração (e um conjunto de 3.003 séries), o M4 possui séries de até 9919 observações e um conjunto de 100.000 séries. Portanto, um universo que pode apresentar maior diversidade nas características das séries que estão distribuídas em diferentes periodicidades e frequências.

Tabela 7 – *Dataset* M4 - Dados estatísticos das séries do *dataset*.

Dados Gerais das Séries				Número de Observações para Treinamento			
Periodicidade	Qtd	Frequência	h	Mín.	Máx	Média	Desvio Padrão
Daily (D)	4227	1	14	93	9919	2357,38	1756,57
Hourly (H)	414	24	48	700	960	853,86	127,95
Monthly (M)	48000	12	18	42	2794	216,30	137,41
Quarterly (Q)	24000	4	8	16	866	92,25	51,13
Weekly (W)	359	1	13	80	2597	1022,04	707,15
Yearly (Y)	23000	1	6	13	835	31,32	24,52
Total:	100000						

Fonte: Autoria própria

4.1.3 *Dataset* M5

O *dataset* M5 (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2020) utilizado neste experimento refere-se ao conjunto de séries temporais disponibilizado para a competição M5, que inclui dados de um total de 42.840 séries temporais (agrupadas hierarquicamente).

Trata-se de mais um conjunto de séries que faz parte das *Makridakis Competitions*, proposta pelo *Makridakis Open Forecasting Center (MOFC)* da Universidade de Nicosia (https://mofc.unic.ac.cy/m5-competition/). Na edição de 2020 (a M5) foi conduzida como uma competição no site da *Kaggle* (https://www.kaggle.com/c/m5-forecasting-accuracy).

Os dados desse *dataset* são referentes a séries temporais de vendas de produtos realizadas por uma grande rede de supermercados norte-americana, em suas unidades dos estados da Califórnia, Texas e Wisconsin relativas a um período de 1941 dias.

Diferentemente de edições anteriores das competições *Makridakis Competitions*, cujos dados das séries consistiam basicamente na lista de observações de cada série (no formato data/observações), na M5 os dados são referentes aos totais vendidos para cada produto, por uma loja, numa determinada data. Além disso, os dados disponibilizados trazem informações sobre os locais de venda e também variáveis explanatórias como preços de vendas, promoções, dados sobre o calendário e datas especiais, que podem interferir no volume vendido em cada loja/região. Outra característica importante, é que as séries desse *dataset* apresentam intermitência dos dados, caracterizada pela possível ocorrência de zeros entre uma observação e outra, relativa à ausência de vendas do produto numa determinada data.

Para a competição, os dados de vendas de produtos por loja, num total de **30.490 séries temporais**, são consolidados de forma hierárquica, agrupando as predições por produto, por categoria, por loja, por estado até completar o total de 42.840 séries originalmente estabelecidas para a competição.

Dessa forma, para este estudo, são de interesse as 30.490 séries que se referem ao nível mais baixo da hierarquia e que representam o total de vendas de um produto agregadas para uma determinada loja. Sobre os dados das séries, para cada produto/série têm-se até 1941 observações relativas às vendas registradas. Para cada série, os 14 registros finais serão reservados para testes, restando, portanto, 1927 observações para o treinamento.

Do ponto de vista de periodicidades e número de observações das séries, a Tabela 8 apresenta os dados estatísticos das séries de M5 consideradas para este experimento. Os valores de horizonte de predição (h) indicados na tabela se referem aos valores convencionados para o experimento para séries de periodicidade Diária e expressam o número de *forecasts* a serem calculados para cada série. Os valores de frequência foram padronizados para todos os *datasets* envolvidos no experimento de acordo com a periodicidade ou tipo da série.

Tabela 8 – *Dataset* M5 - Dados estatísticos das séries do *dataset*.

Dados Gerais das Séries				Número de Observações para Treinamento			
Periodicidade	Qtd	Frequência	h	Mín.	Máx	Média	Desvio Padrão
Diária	30490	1	14	1927	1927	1927	0
Total:	30490						

Fonte: Autoria própria

4.1.4 *Dataset* COVID-19

O *dataset* COVID-19 utilizado nos experimentos deste estudo corresponde a um conjunto de séries temporais disponibilizadas no repositório *COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University*, disponível em https://github.com/CSSEGISandData/COVID-19, (DONG; DU; GARDNER, 2020). Esse conjunto de dados contém informações sobre a pandemia da doença COVID-19, causada pelo coronavirus denominado de SARS-CoV-2, e é o repositório para o *2019 Novel Coronavirus Visual Dashboard* operado pela Universidade Johns Hopkins e suportado pelo *ESRI Living Atlas Team* e *Johns Hopkins University Applied Physics Lab (JHU APL)*.

Do repositório original, foram consideradas neste estudo as tabelas contendo os totais de casos confirmados para 3340 localidades dos Estados Unidos (tabela time_series _covid19_confirmed_US.csv), os totais de falecimentos registrados em decorrência da doença em 3340 localidades norte-americanas (tabela time_series_covid19_deaths_US.csv), os totais de casos confirmados em 266 localidades em nível global (tabela time_series_covid19 _confirmed_global.csv), os totais de falecimentos registrados em decorrência da doença em 266 localidades em nível global (tabela time_series_covid19_deaths_global.csv) e os

totais de casos recuperados em 253 localidades em nível global (tabela time_series_covid19 _recovered_global.csv).

Do ponto de vista de periodicidades e número de observações das séries, a Tabela 9 apresenta os dados estatísticos das séries do *dataset* COVID-19 consideradas para este experimento. Os valores de horizonte de predição (h) indicados na tabela se referem aos valores convencionados para o experimento para séries de periodicidade Diária e expressam o número de *forecasts* a serem calculados para cada série. Os valores de frequência foram padronizados para todos os *datasets* envolvidos no experimento de acordo com a periodicidade ou tipo da série.

Tabela 9 – *Dataset* COVID-19 - Dados estatísticos das séries do *dataset*. Séries com Periodicidade Diária.

Dados Gerais das Séries				Número de Observações para Treinamento			
Dataset	Qtd	Frequência	h	Mín.	Máx	Média	Desvio Padrão
Confirmed US	3340	1	14	224	224	224	0
Deaths US	3340	1	14	224	224	224	0
Confirmed Global	266	1	14	224	224	224	0
Deaths Global	266	1	14	224	224	224	0
Recovered Global	253	1	14	224	224	224	0
Total:	7465						

Fonte: Autoria própria

Para este estudo, os dados foram extraídos do repositório no dia 16/09/2020, com dados referentes ao período de 22/01/2020 a 15/09/2020, totalizando 238 dias. Desse total de dias, os 14 registros finais foram reservados para testes, separando para treinamento informações referentes a 224 dias de observações.

Como os dados contidos nesse *dataset* são acumulados, a característica principal das séries desse conjunto é que elas apresentam tendência de crescimento bem evidente, com períodos de estabilidade, porém sem apresentar diminuições de valores. Com todas as séries com número de observações padronizadas (224 observações), o que se observa é que as localidades que demoraram a registrar a primeira ocorrência da doença podem conter uma grande faixa de valores zerados no início da série.

4.1.5 *Dataset* TSDL

O *dataset* TSDL (*Time Series Data Library*) (HYNDMAN; YANG, 2020) é um conjunto bem diversificado de séries temporais criado por Rob J. Hyndman, professor de Estatística da Universidade Monash, Austrália, autor de diferentes artigos e materiais sobre o tema predição, especialmente no que se refere ao uso de métodos estatísticos.

O *dataset* é composto por séries de diferentes frequências (0.1 a 365), obtidas de várias fontes e representam dados de assuntos como agricultura, ecologia, produção, transporte, entre outros. A Figura 26 apresenta a lista geral das séries que compõem o *dataset* TSDL.

Figura 26 – Lista geral de séries que compõem o dataset TSDL.

Time Series Data Library: 648 time series

	Frequency										
Subject	0.1	0.25	1	4	5	6	12	13	52	365	Total
Agriculture	0	0	37	0	0	0	3	0	0	0	40
Chemistry	0	0	8	0	0	0	0	0	0	0	8
Computing	0	0	6	0	0	0	0	0	0	0	6
Crime	0	0	1	0	0	0	2	1	0	0	4
Demography	1	0	9	2	0	0	3	0	0	2	17
Ecology	0	0	23	0	0	0	0	0	0	0	23
Finance	0	0	23	5	0	0	20	0	2	1	51
Health	0	0	8	0	0	0	6	0	1	0	15
Hydrology	0	0	42	0	0	0	78	1	0	6	127
Industry	0	0	9	0	0	0	2	0	1	0	12
Labour market	0	0	3	4	0	0	17	0	0	0	24
Macroeconomic	0	0	18	33	0	0	5	0	0	0	56
Meteorology	0	0	18	0	0	0	17	0	0	12	47
Microeconomic	0	0	27	1	0	0	7	0	1	0	36
Miscellaneous	0	0	4	0	1	1	3	0	1	0	10
Physics	0	0	12	0	0	0	4	0	0	0	16
Production	0	0	4	14	0	0	28	1	1	0	48
Sales	0	0	10	3	0	0	24	0	9	0	46
Sport	0	1	1	0	0	0	0	0	0	0	2
Transport and tourism	0	0	1	1	0	0	12	0	0	0	14
Tree-rings	0	0	34	0	0	0	1	0	0	0	35
Utilities	0	0	2	1	0	0	8	0	0	0	11
Total	1	1	300	64	1	1	240	3	16	21	648

Fonte: (HYNDMAN; YANG, 2020)

Para este estudo, foram escolhidas as frequências que apresentam maior quantidade de séries (frequências 1, 4 e 12), selecionando desses grupos apenas as séries com mais de 20 observações. Com isso, a Tabela 10 apresenta as estatísticas para as séries selecionadas para o experimento.

Tabela 10 – *Dataset* TSDL - Dados estatísticos das séries do *dataset*, considerando todas as séries originais do conjunto.

Dados Gerais das Séries				Número de Observações para Treinamento			
Frequência original	Qtd	Frequência	h	Mín.	Máx	Média	Desvio Padrão
1	299	1	6	21	39664	885,50	3269,02
4	63	4	8	24	423	123,33	74,00
12	238	12	18	16	2976	385,72	384,55
Total:	600						

Fonte: Autoria própria

Os valores de horizonte de predição (h) indicados na tabela se referem aos valores convencionados para o experimento para séries de periodicidades Diária, Trimestral e

Mensal e expressam o número de *forecasts* a serem calculados para cada série de frequências 1, 4 e 12, respectivamente. Os valores de frequência adotados no processamento são os originais do *dataset* e são compatíveis com os adotados nos outros conjuntos de séries utilizados nos experimentos.

4.2 Experimentos

Esta seção descreve os procedimentos adotados para a realização de experimentos com o método AA-ACF na predição de séries temporais dos *datasets* M3, M4, M5, COVID-19 e TSDL.

4.2.1 Ambiente de Execução

Devido à quantidade de *datasets* e séries temporais envolvidos, os experimentos foram realizados em 3 diferentes ambientes de desenvolvimento devidamente configurados.

Um notebook Lenovo ideapad 3305-14lKB, com processador Intel(R) Core(TM) i5-8250U CPU @1.60GHz 1.80GHz, 8 GB RAM, sistema operacional de 64 bits Windows 10 Home Single Language v. 1909, linguagem R versão 3.6.0, RStudio versão 1.2.1335.

Um desktop iMac, processador Intel(R) Core(TM) i5, 12 GB RAM, sistema operacional macOS High Sierra versão 10.13.6, linguagem R versão 3.4.4, Rstudio versão 1.0.153.

Uma máquina virtual com processador Intel(R) Xeon(R) CPU ES-2660 v4 @ 2.00GHz, 32 GB RAM, sistema operacional de 64 bits Windows 10 Education, linguagem R versão 4.0.2, RStudio versão 1.3.1056.

Quanto ao framework de mineração de fluxo de dados MOA, a versão utilizada nos 3 ambientes de execução foi a versão 2019.05.0.

Os algoritmos estatístico (auto.arima) e de DSM (AdaGrad), bem como os principais pacotes para a linguagem R utilizados na execução do experimentos, estão citados na descrição do método AA-ACF.

Planilhas Microsoft Excel foram criadas para consolidação de dados e geração dos resultados finais do experimento.

4.2.2 Horizonte de Predição (Número de *Forecasts*)

O total de *forecasts* gerados para cada série são estabelecidos de acordo com a periodicidade ou tipo da série em processamento e são definidos para cada série, conforme apresentado em seção anterior que descreve os *datasets* do experimento.

4.2.3 Execução

Para a execução dos experimentos, o método AA-ACF (denominado também como técnica SeleçãoEFusão) foi utilizado de forma a processar as atividades demonstradas no Capítulo 3 deste estudo, porém com adaptações para gerar resultados das execuções das técnicas alternativas do método (uso isolado de auto.arima, uso isolado de AdaGrad, SeleçãoNRegs e Seleção40p) à medida que as predições eram calculadas.

Para a geração de valores para a técnica alternativa Proporção80/20, um programa em R foi gerado de forma a obter o resultado da fusão de predições isoladas de auto.arima e AdaGrad em diferentes proporções, de 0 a 100% de participação de cada algoritmo. Inicialmente, a proporção foi estabelecida como 100% do *forecast* de auto.arima somado a 0% do *forecast* calculado por AdaGrad; em seguida, a proporção foi estabelecida como 90% do *forecast* de auto.arima somado a 10% do *forecast* calculado por AdaGrad e assim sucessivamente, até se chegar ao *forecast* calculado com a proporção de 0% de auto.arima e 100% de AdaGrad. Essa estratégia foi utilizada para poder avaliar para cada *dataset* o valor obtido com a proporção 80% auto.arima e 20% AdaGrad e para identificar outra combinação que pudesse apresentar melhores resultados. De uma forma geral, nenhuma proporção se destacou das demais. Em testes prévios, com subconjuntos pequenos de séries, a proporção 80% auto.arima e 20% AdaGrad sugeria ser uma combinação com melhores resultados e por isso foi selecionada para os experimentos finais.

O processo de seleção de algoritmo foi executado em planilha Excel para os *datasets* utilizando os dados gerados durante os processos de treinamento e predição calculados com auto.arima e AdaGrad, de forma a poder executar diferentes simulações com os valores obtidos.

4.3 Métrica / Protocolo de Avaliação

Para os experimentos descritos neste estudo, a métrica escolhida para a verificação da acurácia do processo de predição (que confronta os valores preditos com os valores reais associados à série) é o sMAPE, acrônimo de *Symmetric Mean Absolute Percentage Error*, também conhecido como *Symmetric* MAPE (ou MAPE simétrico).

O sMAPE é uma métrica de avaliação de erro originalmente proposta em (ARMSTRONG, 1985), utilizada no protocolo de avaliação deste trabalho por ser uma das métricas utilizadas em competições de predição de séries temporais, como as competições M3 e M4, e por permitir a comparação de desempenho de diferentes métodos e *datasets* por se tratar de uma métrica de erro percentual.

Em (MAKRIDAKIS; HIBON, 2000) os autores propõem a seguinte definição

(Equação 4.1):

$$sMAPE = \sum \frac{|X - F|}{(X + F)/2} * 100 \tag{4.1}$$

onde X corresponde ao valor real da observação, e F é o valor predito. E o sMAPE final é definido como a média de todas as diferenças de *forecasts* calculadas para um horizonte de predição.

Em (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018b) os autores apresentam mais claramente como considerar o horizonte de predição no cálculo de sMAPE (Equação 4.2):

$$sMAPE = \frac{2}{k} \sum_{t=1}^{k} \frac{\left|Y_t - \hat{Y}_t\right|}{|Y_t| + \left|\hat{Y}_t\right|} * 100\% \tag{4.2}$$

onde k é o horizonte de predição, Y_t os valores reais, e $\hat{Y}_t$ as predições para um determinado tempo t.

Para este estudo, o cálculo de sMAPE é feito usando a função *smape()* do pacote *Metrics* (HAMNER; FRASCO, 2018) na linguagem R.

Como resultado, a função *smape()* retorna um valor que varia de 0 a 2, onde 0 indica que não há diferença entre o valor predito e o valor real e 2 corresponde ao valor máximo de erro atribuído pela métrica. Ou seja, valores mais próximos de 0 indicam maior acurácia, menor erro de predição.

Do ponto de vista de método, os valores de sMAPE são calculados individualmente para cada série, para cada horizonte de predição. Depois é calculada a média considerando o horizonte total de predição de acordo com a frequência da série. No caso de séries mensais, por exemplo, o horizonte total de predição estabelecido para este estudo é de 18 meses; para séries anuais o horizonte é de 6 anos. O erro de predição obtido por cada método para cada *dataset* é calculado como uma média dos valores de sMAPE individuais obtidos para cada série. Como ***benchmark*** são considerados os valores de sMAPE calculados para as predições realizadas com o método **auto.arima**.

4.4 Resultados dos Experimentos

Esta seção descreve os resultados obtidos pelos experimentos realizados com o método AA-ACF (ou técnica SeleçãoEFusão) na predição de séries temporais dos *datasets* M3, M4, M5, COVID-19 e TSDL.

Os resultados são disponibilizados separadamente por *dataset* e periodicidade das séries, de forma tabular, com o objetivo de apresentar os valores de sMAPE médio obtido

por diferentes técnicas de combinação na geração de predições para as séries do conjunto em análise.

A Tabela 11 descreve o significado das diferentes informações apresentadas nas tabelas de resultados.

Tabela 11 – Tabela explicativa das informações apresentadas nas tabelas com resultados dos experimentos.

Campo	Descrição
auto.arima (Método)	Erros de predições efetuadas com o uso isolado do método auto.arima
AdaGrad (Método)	Erros de predições efetuadas com o uso isolado do método AdaGrad
SeleçãoNRegs (Método)	Erros de predições obtidos pelo método com seleção feita com base no erro de treinamento calculado considerando os n últimos registros (n=h=*número de forecasts da série*)
Seleção40p (Método)	Erros de predições obtidos pelo método com seleção feita com base no erro de treinamento calculado considerando os 40% dos registros finais de treinamento
SeleçãoEFusão (Método)	Corresponde aos valores de erros de predição obtidos com o método AA-ACF que, além de fazer a seleção conforme SeleçãoNRegs, efetua fusão de *forecasts* de auto.arima e AdaGrad para os casos em que AdaGrad seja o algoritmo selecionado para a série
%ARI %ADG (Método)	Apresenta resultados obtidos com o uso de fusão de *forecasts* de auto.arima e AdaGrad na proporção indicada na tabela
Total de séries auto.arima	Corresponde ao total de séries selecionadas para uso de auto.arima para o método em questão
Total de séries AdaGrad	Corresponde ao total de séries selecionadas para uso de AdaGrad para o método em questão
sMAPE médio	Apresenta o erro médio de teste calculado para todas as séries do conjunto considerando as predições efetuadas para todo o horizonte de predição
Ganho %	Corresponde ao ganho percentual obtido com o método em questão, calculado em relação ao erro apresentado por auto.arima usado isoladamente

Fonte: Autoria própria

As tabelas de resultados apresentam o ganho obtido pelo uso de diferentes métodos em relação ao uso isolado do algoritmo auto.arima. O ganho em questão é calculado conforme a Equação 4.3 e leva em consideração o $sMAPE_{método}$ (sMAPE de teste obtido pelo método em análise) e o $sMAPE_{auto.arima}$ (sMAPE de teste obtido pelo uso isolado de auto.arima).

$$Ganho = -\left(\left(\frac{sMAPE_{metodo}}{sMAPE_{auto.arima}}\right) - 1\right) * 100(\%) \tag{4.3}$$

O método AA-ACF gera os *forecasts* para cada série na quantidade estabelecida de acordo com sua periodicidade. Conforme explicado anteriormente, para séries mensais, por exemplo, o método gera 18 *forecasts* individuais, um para cada horizonte de predição (h1 a

h18). Para efeitos de comparação, esta seção apresenta os resultados médios obtidos para todo o horizonte de predição, uma vez que o objetivo principal do estudo é o de identificar o comportamento do método em problemas de multisséries, e verificar o desempenho do mesmo para o horizonte completo de predição.

A título de exemplo, no entanto, para as séries Mensais de M3, serão apresentados os resultados de sMAPE médios obtidos para cada horizonte de predição, de forma a explicar que é a média dos sMAPEs individuais que estabelece o valor de sMAPE médio calculado para cada método aplicado em cada *dataset*.

4.4.1 *Dataset* M3

Esta seção apresenta os resultados obtidos com os experimentos realizados com as séries do *dataset* M3.

A Tabela 12 apresenta os resultados obtidos com o processamento das séries Mensais de M3. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 1428 séries mensais e nos *forecasts* gerados para 18 meses (h1 a h18).

Tabela 12 – *Dataset* M3 - Resultados do processamento de séries Mensais.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	1428	0	14,918%	
AdaGrad	0	1428	15,586%	-4,476%
SeleçãoNRegs	1180	248	14,420%	3,339%
Seleção40p	1247	181	14,411%	3,397%
SeleçãoEFusão	1180	248	14,340%	3,875%
	Proporção			
%ARI %ADG	80%	20%	14,325%	3,979%
%ARI %ADG	60%	40%	14,143%	5,198%

Fonte: Autoria própria

Analisando o desempenho da técnica de SeleçãoEFusão (que corresponde aos valores obtidos com o método AA-ACF em sua configuração *default*), observa-se que o método proposto foi capaz de apresentar um ganho positivo de 3,875% em relação aos valores preditos por auto.arima usado isoladamente. Com exceção do uso isolado de AdaGrad que apresentou ganho negativo, as demais técnicas foram capazes de superar os valores de auto.arima.

A título de detalhamento (e de exemplo), a Tabela 13 apresenta os valores de sMAPE calculados para cada um dos horizontes de predição (no caso h1 a h18). São esses valores que são considerados no cômputo do valor de sMAPE médio apresentado

anteriormente na Tabela 12. Ao analisar os valores de cada horizonte de predição (h1, h2, h3, ...), é possível verificar para cada mês o desempenho individual de cada técnica utilizada. Pode-se observar, por exemplo, que para todos os meses, os valores obtidos pela técnica de SeleçãoEFusão são menores que os apresentados por auto.arima.

Tabela 13 – *Dataset* M3 - Resultados do processamento de séries Mensais, apresentando valores de sMAPE obtidos para cada horizonte de predição de h1 a h18.

	sMAPE calculado para cada horizonte de predição								
Método	h1	h2	h3	h4	h5	h6	h7	h8	h9
auto.arima	0,114	0,115	0,125	0,134	0,126	0,126	0,138	0,135	0,136
AdaGrad	0,128	0,125	0,128	0,140	0,124	0,130	0,138	0,137	0,135
SeleçãoNregs	0,112	0,110	0,121	0,131	0,121	0,122	0,133	0,130	0,131
Seleção40p	0,112	0,111	0,121	0,131	0,121	0,123	0,134	0,130	0,131
SeleçãoEFusão	0,112	0,111	0,121	0,131	0,120	0,122	0,134	0,130	0,131
80%ARI 20%ADG	0,113	0,113	0,121	0,131	0,119	0,122	0,134	0,130	0,130
60%ARI 40%ADG	0,115	0,113	0,120	0,130	0,115	0,121	0,131	0,128	0,127
Método	h10	h11	h12	h13	h14	h15	h16	h17	h18
auto.arima	0,139	0,144	0,140	0,166	0,169	0,185	0,198	0,194	0,200
AdaGrad	0,154	0,149	0,150	0,182	0,187	0,196	0,202	0,200	0,198
SeleçãoNregs	0,137	0,140	0,138	0,159	0,166	0,182	0,190	0,184	0,189
Seleção40p	0,138	0,139	0,137	0,158	0,165	0,180	0,188	0,184	0,190
SeleçãoEFusão	0,136	0,139	0,136	0,158	0,161	0,177	0,189	0,183	0,191
80%ARI 20%ADG	0,136	0,139	0,135	0,162	0,162	0,175	0,188	0,182	0,188
60%ARI 40%ADG	0,137	0,137	0,134	0,162	0,160	0,173	0,184	0,178	0,184

Fonte: Autoria própria

A Tabela 14 apresenta os resultados obtidos com o processamento das séries Anuais de M3. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 645 séries anuais e nos *forecasts* gerados para 6 anos (h1 a h6).

Para as séries anuais não foi possível observar nenhuma técnica capaz de superar os valores obtidos por auto.arima. O baixo desempenho de AdaGrad, que mostra um ganho negativo de -87,949% em relação aos valores de auto.arima impactam bastante sobre as demais técnicas que são dependentes do resultado isolado desse método de DSM e isso pode estar relacionado ao baixo comprimento das séries desse conjunto, visto que as séries anuais de M3 possuem de 14 a 41 observações.

A Tabela 15 apresenta os resultados obtidos com o processamento das séries Trimestrais de M3. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 756 séries trimestrais e nos *forecasts* gerados para 8 trimestres (h1 a h8).

Tabela 14 – *Dataset* M3 - Resultados do processamento de séries Anuais.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	645	0	17,104%	
AdaGrad	0	645	32,147%	-87,949%
SeleçãoNRegs	615	30	17,273%	-0,991%
Seleção40p	629	16	17,229%	-0,732%
SeleçãoEFusão	615	30	17,157%	-0,309%
	Proporção			
%ARI %ADG	80%	20%	17,666%	-3,283%
%ARI %ADG	90%	10%	17,150%	-0,271%

Fonte: Autoria própria

Tabela 15 – *Dataset* M3 - Resultados do processamento de séries Trimestrais.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	756	0	10,011%	
AdaGrad	0	756	15,652%	-56,349%
SeleçãoNRegs	681	75	10,055%	-0,437%
Seleção40p	726	30	9,971%	0,397%
SeleçãoEFusão	681	75	9,973%	0,380%
	Proporção			
%ARI %ADG	80%	20%	10,365%	-3,533%
%ARI %ADG	90%	10%	10,087%	-0,755%

Fonte: Autoria própria

Para o conjunto de séries trimestrais, é observado um ganho positivo de 0,380% da técnica SeleçãoEFusão em relação aos valores de auto.arima. A técnica de Seleção40p apresentou um ganho ligeiramente maior (0,397%), que sugere que a análise de uma porção maior de dados de treinamento é melhor para a seleção do método mais adequado para a série. Para processos em *batch* essa técnica poderia ser considerada como uma opção, no entanto, pensando na aplicação para um cenário de fluxo contínuo de dados, pode ser inviável contabilizar erro sobre um número variável de observações uma vez que a técnica Seleção40p pressupõe a separação de 40% dos dados de treinamento para o cálculo de erro nessa etapa.

A Tabela 16 apresenta os resultados obtidos com o processamento das séries do tipo Outras de M3. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento de 174 séries e nos *forecasts* gerados para 8 períodos (h1 a h8).

Para as séries do tipo Outras, a técnica de SeleçãoEFusão apresentou um ganho

Tabela 16 – *Dataset* M3 - Resultados do processamento de séries do tipo Outras.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	174	0	4,513%	
AdaGrad	0	174	5,454%	-20,850%
SeleçãoNRegs	151	23	4,391%	2,711%
Seleção40p	169	5	4,517%	-0,091%
SeleçãoEFusão	151	23	4,419%	2,078%
	Proporção			
%ARI %ADG	80%	20%	4,476%	0,821%

Fonte: Autoria própria

positivo de 2,078% em relação a auto.arima. Esse valor, no entanto, foi superado pelo ganho apresentado pela técnica SeleçãoNRegs que não faz uso da etapa de fusão na definição dos *forecasts* finais das séries. O fato de esses valores terem sido melhores que a técnica baseada em fusão sugere que os valores de AdaGrad utilizado isoladamente apresentaram resultados melhores que os valores fundidos com auto.arima nas séries em que o processo de seleção tenha recomendado o algoritmo AdaGrad para as séries.

4.4.2 Dataset M4

Esta seção apresenta os resultados obtidos com os experimentos realizados com as séries do *dataset* M4.

A Tabela 17 apresenta os resultados obtidos com o processamento das séries Diárias de M4. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 4227 séries diárias e nos *forecasts* gerados para 14 dias (h1 a h14).

Tabela 17 – *Dataset* M4 - Resultados do processamento de séries Diárias.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	4227	0	3,186%	
AdaGrad	0	4227	4,063%	-27,518%
SeleçãoNregs	3680	547	3,200%	-0,449%
Seleção40p	4210	17	3,182%	0,116%
SeleçãoEFusão	3680	547	3,145%	1,288%
	Proporção			
%ARI %ADG	80%	20%	3,115%	2,243%

Fonte: Autoria própria

Para as séries diárias, a técnica de SeleçãoEFusão foi capaz de apresentar um

ganho positivo de 1,288% em relação ao erro apresentado por auto.arima. A combinação proporcional de auto.arima (80%) e AdaGrad (20%), por sua vez, foi capaz de apresentar um ganho de 2,243%, superior ao apresentado pela técnica de SeleçãoEFusão. O uso de combinação baseada em proporção é dificultado no aspecto de escolha dos percentuais adequados a usar na combinação de diferentes *forecasts* em diferentes conjuntos de séries. Em todo caso, é um método usual e, por isso, foi considerado em SeleçãoEFusão com aplicação limitada às séries que tenham sido atribuídas a AdaGrad, sempre num percentual de 50% para cada método.

A Tabela 18 apresenta os resultados obtidos com o processamento das séries Horárias de M4. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 414 séries horárias e nos *forecasts* gerados para 48 horas (h1 a h48).

Tabela 18 – *Dataset* M4 - Resultados do processamento de séries Horárias.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	414	0	14,090%	
AdaGrad	0	414	27,325%	-93,932%
SeleçãoNregs	395	19	14,485%	-2,803%
Seleção40p	409	5	14,221%	-0,931%
SeleçãoEFusão	395	19	14,242%	-1,083%
	Proporção			
%ARI %ADG	80%	20%	15,378%	-9,142%
%ARI %ADG	90%	10%	14,272%	-1,293%

Fonte: Autoria própria

Analisando os resultados obtidos para as séries horárias é possível observar que nenhuma técnica apresentou ganhos positivos em relação a auto.arima.

A Tabela 19 apresenta os resultados obtidos com o processamento das séries Mensais de M4. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 48000 séries mensais e nos *forecasts* gerados para 18 meses (h1 a h18).

A técnica de SeleçãoEFusão apresentou um ganho de 0,499% em relação ao uso individual de auto.arima para as séries mensais. Já a combinação em diferentes proporções de *forecasts* de auto.arima e AdaGrad alcançou ganho de 2,882% (para 80% auto.arima e 20% AdaGrad) e de 3,230% (para 70% auto.arima e 30% AdaGrad).

A Tabela 20 apresenta os resultados obtidos com o processamento das séries Trimestrais de M4. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 24000 séries trimestrais e nos

Tabela 19 – *Dataset* M4 - Resultados do processamento de séries Mensais.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	48000	0	13,495%	
AdaGrad	0	48000	14,791%	-9,602%
SeleçãoNregs	42311	5689	13,554%	-0,438%
Seleção40p	46029	1971	13,475%	0,146%
SeleçãoEFusão	42311	5689	13,427%	0,499%
	Proporção			
%ARI %ADG	80%	20%	13,106%	2,882%
%ARI %ADG	70%	30%	13,059%	3,230%

Fonte: Autoria própria

forecasts gerados para 8 trimestres (h1 a h8).

Para as séries trimestrais, a técnica de SeleçãoEFusão mostrou novamente ganho positivo em relação ao uso isolado de auto.arima. Um ganho no valor de 0,384%.

Tabela 20 – *Dataset* M4 - Resultados do processamento de séries Trimestrais.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	24000	0	10,420%	
AdaGrad	0	24000	12,386%	-18,872%
SeleçãoNregs	19905	4095	10,543%	-1,179%
Seleção40p	22919	1081	10,436%	-0,151%
SeleçãoEFusão	19905	4095	10,380%	0,384%
	Proporção			
%ARI %ADG	80%	20%	10,317%	0,989%

Fonte: Autoria própria

A Tabela 21 apresenta os resultados obtidos com o processamento das séries Semanais de M4. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 359 séries semanais e nos *forecasts* gerados para 13 semanas (h1 a h13).

Para as séries semanais de M4, nenhuma das técnicas padrões obteve ganho positivo em relação ao uso isolado de auto.arima. Analisando diferentes combinação de *forecasts* de auto.arima e AdaGrad foi possível observar um ganho positivo de 0,269% ao usar a proporção de 90% auto.arima e 10% AdaGrad na definição de *forecasts* para as séries. Esse tipo de análise serve, no entanto, apenas como caráter comparativo, visto que valores de proporção num método em uso contínuo (em produção) devem ser estabelecidos de antemão ao início de processamento (a menos que o método consiga fazer ajustes nas

Tabela 21 – *Dataset* M4 - Resultados do processamento de séries Semanais.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	359	0	8,591%	
AdaGrad	0	359	11,440%	-33,158%
SeleçãoNregs	276	83	9,513%	-10,734%
Seleção40p	346	13	8,622%	-0,357%
SeleçãoEFusão	276	83	8,889%	-3,462%
	Proporção			
%ARI %ADG	80%	20%	8,630%	-0,451%
%ARI %ADG	90%	10%	8,568%	0,269%

Fonte: Autoria própria

proporções ideais durante o processo de predição/teste).

A Tabela 22 apresenta os resultados obtidos com o processamento das séries Anuais de M4. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 23000 séries anuais e nos *forecasts* gerados para 6 anos (h1 a h6).

Tabela 22 – *Dataset* M4 - Resultados do processamento de séries Anuais.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	23000	0	15,156%	
AdaGrad	0	23000	21,407%	-41,241%
SeleçãoNregs	20239	2761	15,267%	-0,730%
Seleção40p	21816	1184	15,171%	-0,101%
SeleçãoEFusão	20239	2761	15,088%	0,449%
	Proporção			
%ARI %ADG	80%	20%	15,000%	1,027%
%ARI %ADG	90%	10%	14,949%	1,364%

Fonte: Autoria própria

Para séries com periodicidade anual, a técnica de SeleçãoEFusão, principal interesse deste estudo, conseguiu apresentar ganho positivo em relação aos valores apresentados por auto.arima usado isoladamente.

4.4.3 *Dataset* M5

Esta seção apresenta os resultados obtidos com os experimentos realizados com as séries do *dataset* M5.

A Tabela 23 apresenta os resultados obtidos com o processamento das 30490 séries Diárias de M5. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento de todas as séries do conjunto e nos *forecasts* gerados para 14 dias (h1 a h14).

Tabela 23 – *Dataset* M5 - Resultados do processamento de séries Diárias.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	30490	0	136,878%	
AdaGrad	0	30490	122,966%	10,164%
SeleçãoNregs	14561	15929	120,625%	11,874%
Seleção40p	7833	22657	121,531%	11,212%
SeleçãoEFusão	14561	15929	137,509%	-0,461%
	Proporção			
%ARI %ADG	80%	20%	137,958%	-0,789%
%ARI %ADG	90%	10%	137,884%	-0,735%

Fonte: Autoria própria

Para as séries de M5, foi possível observar um ganho positivo bem expressivo com o uso da técnica de SeleçãoNRegs, que faz a seleção de algoritmo com base no erro de treinamento, porém não faz uso da fusão de *forecasts* de auto.arima e AdaGrad tal como a técnica de SeleçãoEFusão. O ganho com a técnica SeleçãoNRegs foi de 11,874%. A seleção feita com base em 40% dos registros finais de treinamento (Seleção40p) também apresentou um ganho bem elevado, igual a 11,212%. Os bons resultados dessas duas técnicas pode ser explicado pelo fato de o uso isolado de AdaGrad ter apresentado também alto valor de ganho (10,164%). Em contrapartida, o uso de fusão de forecasts nesse cenário não foi capaz de apresentar resultados positivos.

4.4.4 *Dataset* COVID-19

Esta seção apresenta os resultados obtidos com os experimentos realizados com as séries do *dataset* COVID-19. Para esse *dataset*, todas as séries são diárias e os valores de sMAPE médio apresentados nas tabelas correspondem ao valor médio obtido como resultado do processamento de todas as séries do conjunto e nos *forecasts* gerados para 14 dias (h1 a h14).

A Tabela 24 apresenta resultados de processamentos das 3340 séries do subconjunto *Confirmed US*.

Para as séries de casos confirmados (*Confirmed US*) as técnicas SeleçãoNRegs e Seleção40p foram as que apresentaram maiores ganhos positivos em relação a auto.arima (5,206% e 4,959%, respectivamente). A fusão de *forecasts* utilizada pela técnica de Se-

Tabela 24 – *Dataset* COVID-19 - Resultados do processamento de séries do *dataset Confirmed US*.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	3340	0	5,848%	
AdaGrad	0	3340	8,011%	-36,991%
SeleçãoNregs	3160	180	5,543%	5,206%
Seleção40p	3261	79	5,558%	4,959%
SeleçãoEFusão	3160	180	5,853%	-0,088%
	Proporção			
%ARI %ADG	80%	20%	6,513%	-11,381%
%ARI %ADG	90%	10%	6,438%	-10,091%

Fonte: Autoria própria

leçãoEFusão, que combina os *forecasts* de auto.arima e AdaGrad na proporção de 50% para cada algoritmo apresentou resultado negativo, no valor de -0,088%, sugerindo que os valores de predições de auto.arima estão mais distantes do valor real nas séries selecionadas para AdaGrad.

A Tabela 25 apresenta resultados de processamentos das 3340 séries do subconjunto *Deaths US* .

Tabela 25 – *Dataset* COVID-19 - Resultados do processamento de séries do *dataset Deaths US*.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	3340	0	11,220%	
AdaGrad	0	3340	12,698%	-13,173%
SeleçãoNregs	3105	235	10,499%	6,419%
Seleção40p	3210	130	10,571%	5,780%
SeleçãoEFusão	3105	235	11,211%	0,079%
	Proporção			
%ARI %ADG	80%	20%	11,737%	-4,611%
%ARI %ADG	90%	10%	11,642%	-3,768%

Fonte: Autoria própria

Para o subconjunto *Deaths US*, o método SeleçãoEFusão conseguiu apresentar um ganho positivo em relação a auto.arima no valor de 0,079%, porém bem inferior aos obtidos por SeleçãoNRegs e Seleção40p.

A Tabela 26 apresenta resultados de processamento das 266 séries do subconjunto *Confirmed GLOBAL*.

Tabela 26 – *Dataset* COVID-19 - Resultados do processamento de séries do *dataset Confirmed GLOBAL*.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	266	0	2,783%	
AdaGrad	0	266	6,760%	-142,914%
SeleçãoNregs	250	16	2,780%	0,107%
Seleção40p	263	3	2,836%	-1,908%
SeleçãoEFusão	250	16	2,779%	0,147%
	Proporção			
%ARI %ADG	80%	20%	3,121%	-12,144%
%ARI %ADG	90%	10%	2,900%	-4,220%

Fonte: Autoria própria

Em relação aos casos confirmados globais (*Confirmed Global*), a SeleçãoEFusão apresentou ganho positivo de 0,147%, maior que o obtido com outras técnicas.

A Tabela 27 apresenta resultados de processamento das 266 séries do subconjunto *Deaths GLOBAL*.

Tabela 27 – *Dataset* COVID-19 - Resultados do processamento de séries do *dataset Deaths GLOBAL*.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	266	0	3,671%	
AdaGrad	0	266	6,557%	-78,619%
SeleçãoNregs	254	12	3,704%	-0,900%
Seleção40p	262	4	3,655%	0,425%
SeleçãoEFusão	254	12	3,675%	-0,101%
	Proporção			
%ARI %ADG	80%	20%	4,157%	-13,243%
%ARI %ADG	90%	10%	3,940%	-7,342%

Fonte: Autoria própria

Para o subconjunto *Deaths Global*, a técnica que apresentou melhores resultados foi a de Seleção40p, com ganho de 0,425%.

A Tabela 28 apresenta resultados de processamento das 253 séries do subconjunto *Recovered GLOBAL*.

Em relação ao número de recuperados globais (*Recovered Global*), a técnica SeleçãoEFusão conseguiu apresentar ganho positivo de 1,363% em relação a auto.arima. Outras técnicas que apresentaram bons resultados, superiores inclusive aos obtidos com a técnica

Tabela 28 – *Dataset* COVID-19 - Resultados do processamento de séries do *dataset Recovered GLOBAL*.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	253	0	4,633%	
AdaGrad	0	253	8,349%	-80,198%
SeleçãoNregs	232	21	4,518%	2,479%
Seleção40p	238	15	4,275%	7,736%
SeleçãoEFusão	232	21	4,570%	1,363%
	Proporção			
%ARI %ADG	80%	20%	6,245%	-34,790%
%ARI %ADG	90%	10%	6,146%	-32,659%

Fonte: Autoria própria

de fusão, foram as técnicas de SeleçãoNRegs (2,479%) e Seleção40p (7,736%). O resultado de Seleção40p sugere que observar uma quantidade maior de registros de treinamento auxiliou na identificação das séries melhor processadas com AdaGrad nesse conjunto.

4.4.5 *Dataset* TSDL

Esta seção apresenta os resultados obtidos com os experimentos realizados com as séries do *dataset* TSDL.

A Tabela 29 apresenta os resultados obtidos com o processamento das séries de TSDL com Frequência 1. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 299 séries e nos 6 *forecasts* gerados (para h1 a h6) para cada série.

Tabela 29 – Dataset TSDL - Resultados do processamento de séries de Frequência 1.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	299	0	34,071%	
AdaGrad	0	299	36,697%	-7,706%
SeleçãoNregs	213	86	33,704%	1,077%
Seleção40p	274	25	33,283%	2,313%
SeleçãoEFusão	213	86	32,188%	5,528%
	Proporção			
%ARI %ADG	80%	20%	33,141%	2,731%

Fonte: Autoria própria

Para as séries de Frequência 1, a técnica de SeleçãoEFusão foi a que apresentou maior ganho positivo (5,528%), conseguindo superar os ganhos obtidos pelas demais

técnicas. Analisando os valores de AdaGrad usado isoladamente, observa-se que o ganho obtido por esse algoritmo DSM é negativo, porém quando as séries atribuídas a AdaGrad são melhor identificadas pelo processo de seleção (de forma que os *forecasts* calculados por AdaGrad sejam melhores que os de auto.arima), o ganho positivo é alcançado.

A Tabela 30 apresenta os resultados obtidos com o processamento das séries de TSDL com Frequência 4. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 63 séries e nos 8 *forecasts* gerados (para h1 a h8) para cada série.

Tabela 30 – Dataset TSDL - Resultados do processamento de séries de Frequência 4.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	63	0	22,039%	
AdaGrad	0	63	27,841%	-26,330%
SeleçãoNregs	55	8	22,896%	-3,892%
Seleção40p	60	3	22,759%	-3,268%
SeleçãoEFusão	55	8	19,299%	12,429%
	Proporção			
%ARI %ADG	80%	20%	22,782%	-3,371%
%ARI %ADG	90%	10%	22,298%	-1,178%

Fonte: Autoria própria

Para as séries de Frequência 4, também foi possível observar um ganho positivo nos valores apresentados pela técnica SeleçãoEFusão, que obteve ganho de 12,429%; as demais técnicas, por sua vez, apresentaram valores negativos. Isso pode indicar que a fusão de *forecasts* de auto.arima e AdaGrad utilizada por SeleçãoEFusão pode ter auxiliado esse método a apresentar um ganho que se destacou dos apresentados pelas demais técnicas.

A Tabela 31 apresenta os resultados obtidos com o processamento das séries de TSDL com Frequência 12. Os valores de sMAPE médio apresentados nessa tabela correspondem ao valor médio obtido como resultado do processamento das 238 séries e nos 18 *forecasts* gerados (para h1 a h18) para cada série.

Analisando os resultados do processamento das séries com Frequência 12, é possível observar que o método SeleçãoEFusão (ganho = 8,307%) foi o que apresentou maior ganho positivo em relação a auto.arima quando comparado aos resultados dos demais métodos, que também apresentaram ganhos bem expressivos (com exceção de AdaGrad utilizado isoladamente, que obteve ganho negativo de -3,195%).

Esses resultados demonstram que o uso combinado de técnicas é capaz, em muitos casos, de possibilitar ganhos superiores aos obtidos pelo uso de uma outra técnica de forma individual.

Tabela 31 – Dataset TSDL - Resultados do processamento de séries de Frequência 12.

	Total de Séries			
Método	auto.arima	AdaGrad	sMAPE médio	Ganho %
auto.arima	238	0	44,038%	
AdaGrad	0	238	45,445%	-3,195%
SeleçãoNregs	162	76	41,474%	5,823%
Seleção40p	173	65	41,354%	6,095%
SeleçãoEFusão	162	76	40,380%	8,307%
	Proporção			
%ARI %ADG	80%	20%	42,427%	3,660%
%ARI %ADG	50%	50%	41,717%	5,272%

Fonte: Autoria própria

Esta seção apresentou, separadamente, os resultados obtidos pelo processamento de cada *dataset* e suas séries de diferentes periodicidades. Na próxima seção esses resultados serão comparados entre todos de forma a verificar o método capaz de se destacar dos demais quando aplicado em cenários de múltiplos *datasets*.

4.5 Análise/Discussão dos Resultados

Nesta seção, os resultados obtidos pelo processamento individual dos *datasets* M3, M4, M5, COVID-19 e TSDL são analisados em conjunto de forma a avaliar o desempenho do método AA-ACF (usando a técnica SeleçãoEFusão) em comparação aos resultados das técnicas alternativas avaliadas (uso isolado de auto.arima e AdaGrad, SeleçãoNRegs, Seleção40p, Proporção80/20).

Nas tabelas comparativas de ganhos, a denominação das técnicas pode aparecer de forma abreviada, de forma a otimizar o *layout* de apresentação, com a seguinte convenção: NRegs (SeleçãoNRegs), 40p (Seleção40p), Fusão (SeleçãoEFusão), % (80/20) (Proporção80/20).

Nas subseções a seguir são apresentadas análises considerando diferentes horizontes de predição (h=original, h=6, h=1) e, ao final, uma discussão geral sobre os resultados obtidos.

4.5.1 Análise de Ganhos Considerando Horizontes de Predição Originais (h=6 a h=48)

A Tabela 32 apresenta os ganhos para horizontes de predição originais obtidos pela utilização das técnicas para os diferentes *datasets* do experimento, considerando os erros de

forecasts obtidos para o horizonte completo de predição. Ou seja, levam em consideração os erros médios observados para o horizonte de predição padronizado de acordo com a periodicidade da série (podendo variar de 6, para séries anuais, a 48 para séries horárias). A última linha da Tabela 32 mostra os valores de ganho médio obtidos pelas diferentes técnicas considerando todos os *datasets*/periodicidades que participaram do experimento.

Tabela 32 – Tabela de ganhos percentuais obtidos em relação a auto.arima pelo uso de diferentes técnicas de seleção e/ou combinação. Calculados para os horizontes de predição originais.

Dataset	Tipo Série	Qtd	h	NRegs	40p	Fusão	% (80/20)	% (ARI/ADG)
M3	Mensais	1428	18	3,339%	3,397%	3,875%	3,979%	5,198% (60/40)
M3	Anuais	645	6	-0,991%	-0,732%	-0,309%	-3,283%	-0,271% (90/10)
M3	Trimestrais	756	8	-0,437%	0,397%	0,380%	-3,533%	-0,755% (90/10)
M3	Outras	174	8	2,711%	-0,091%	2,078%	0,821%	0,821% (80/20)
M4	Diárias	4227	14	-0,449%	0,116%	1,288%	2,243%	2,243% (80/20)
M4	Horárias	414	48	-2,803%	-0,931%	-1,083%	-9,142%	-1,293% (90/10)
M4	Mensais	48000	18	-0,438%	0,146%	0,499%	2,882%	3,230% (70/30)
M4	Trimestrais	24000	8	-1,179%	-0,151%	0,384%	0,989%	0,989% (80/20)
M4	Semanais	359	13	-10,734%	-0,357%	-3,462%	-0,451%	0,269% (90/10)
M4	Anuais	23000	6	-0,730%	-0,101%	0,449%	1,027%	1,364% (90/10)
M5	Diárias	30490	14	11,874%	11,212%	-0,461%	-0,789%	-0,735% (90/10)
COVID-19	*Confirmed US*	3340	14	5,206%	4,959%	-0,088%	-11,381%	-10,091% (90/10)
COVID-19	*Deaths US*	3340	14	6,419%	5,780%	0,079%	-4,611%	-3,768% (90/10)
COVID-19	*Confirmed Global*	266	14	0,107%	-1,908%	0,147%	-12,144%	-4,220% (90/10)
COVID-19	*Deaths Global*	266	14	-0,900%	0,425%	-0,101%	-13,243%	-7,342% (90/10)
COVID-19	*Recovered Global*	253	14	2,479%	7,736%	1,363%	-34,790%	-32,659% (90/10)
TSDL	Frequência 1	299	6	1,077%	2,313%	5,528%	2,731%	2,731% (80/20)
TSDL	Frequência 4	63	8	-3,892%	-3,268%	12,429%	-3,371%	-1,178% (90/10)
TSDL	Frequência 12	238	18	5,823%	6,095%	8,307%	3,660%	5,272% (50/50)
		GANHO MÉDIO:		0,87%	1,84%	1,65%	-4,13%	

Fonte: Autoria própria

Com o objetivo de avaliar a existência de diferenças entre os resultados e identificar o método com maior destaque no processo de predição, diferentes análises, testes estatísticos não-paramétricos e *post-hoc tests* foram utilizados conforme proposto por Demšar (2006). Para isso, foram considerados os valores de sMAPE de teste calculados para os *datasets* com base nas predições geradas pelas diferentes técnicas avaliadas. Tais valores são apresentados na Tabela 33, que mostra os valores calculados para os horizontes de predição originais (h = originais).

Considerando os valores de sMAPE obtidos pelos diferentes métodos para a predição para todo o horizonte, diferentes análises podem ser efetuadas. Uma abordagem é o estabelecimento de um ***ranking* com base no valor de sMAPE de teste obtido por**

Tabela 33 – Tabela de sMAPE de teste obtidos pelo uso de diferentes técnicas de seleção e/ou combinação. Valores de sMAPE calculados para todas as séries de cada *dataset* e horizontes de predição originais.

Dataset	auto.arima	AdaGrad	NRegs	40p	Fusão	% (80/20)
M3 Mensais	0,14918	0,15586	0,1442	0,14411	0,1434	0,14325
M3 Anuais	0,17104	0,32147	0,17273	0,17229	0,17157	0,17666
M3 Trimestrais	0,10011	0,15652	0,10055	0,09971	0,09973	0,10365
M3 Outras	0,04513	0,05454	0,04391	0,04517	0,04419	0,04476
M4 Diárias	0,03186	0,04063	0,032	0,03182	0,03145	0,03115
M4 Horárias	0,1409	0,27325	0,14485	0,14221	0,14242	0,15378
M4 Mensais	0,13495	0,14791	0,13554	0,13475	0,13427	0,13106
M4 Trimestrais	0,10420	0,12386	0,10543	0,10436	0,1038	0,10317
M4 Semanais	0,08591	0,1144	0,09513	0,08622	0,08889	0,0863
M4 Anuais	0,15156	0,21407	0,15267	0,15171	0,15088	0,15
M5 Diárias	1,36878	1,22966	1,20625	1,21531	1,37509	1,37958
COVID-19 *Confirmed US*	0,05848	0,08011	0,05543	0,05558	0,05853	0,06513
COVID-19 *Deaths US*	0,1122	0,12698	0,10499	0,10571	0,11211	0,11737
COVID-19 *Confirmed GLOBAL*	0,02783	0,0676	0,0278	0,02836	0,02779	0,03121
COVID-19 *Deaths GLOBAL*	0,03671	0,06557	0,03704	0,03655	0,03675	0,04157
COVID-19 *Recovered GLOBAL*	0,04633	0,08349	0,04518	0,04275	0,0457	0,06245
TSDL Frequência 1	0,34071	0,36697	0,33704	0,33283	0,32188	0,33141
TSDL Frequência 4	0,22039	0,27841	0,22896	0,22759	0,19299	0,22782
TSDL Frequência 12	0,44038	0,45445	0,41474	0,41354	0,4038	0,42427

Fonte: Autoria própria

cada método para cada *dataset*, atribuindo 1 para a técnica que tenha apresentado menor valor de erro médio para um dataset e 6 para aquela que tenha apresentado o maior erro. A faixa de pontuação varia de 1 a 6 pois foram efetuadas predições com 6 diferentes técnicas. Uma vez estabelecido o *ranking* por *dataset*, um *ranking* geral é calculado com o objetivo de identificar o método que obteve melhor posição em mais ocorrências. A Tabela 34 apresenta o resultado da análise de *ranking* com base nos sMAPEs de teste para horizontes de predição originais (Tabela 33).

Analisando o número de ocorrências em que cada técnica alcançou o menor valor de sMAPE médio de teste para todo o horizonte de predição para todas as séries de um determinado *dataset*, obtém-se o *ranking* (Tabela 35) que indica que a técnica de SeleçãoEFusão foi a que apresentou, em mais experimentos, o menor valor de sMAPE médio de teste para todas as séries dos *datasets* avaliados. Nesse método, a seleção de algoritmo é feita com base no erro de treinamento (sMAPE de treinamento) calculado para os n registros finais da série (onde n corresponde ao horizonte de predição h estabelecido para a periodicidade da série), seguido de fusão dos valores de *forecasts* de auto.arima e AdaGrad nos registros cuja seleção tenha indicado o método AdaGrad.

Tabela 34 – Tabela de ranking de valores de sMAPE de teste obtidos pelo uso de diferentes técnicas de seleção e/ou combinação. Valores de sMAPE calculados para os horizontes de predição originais. *Ranking* estabelecido por *dataset*.

Dataset	auto.arima	AdaGrad	NRegs	40p	Fusão	%(80/20)
M3 Mensais	5	6	4	3	2	1
M3 Anuais	1	6	4	3	2	5
M3 Trimestrais	3	6	4	1	2	5
M3 Outras	4	6	1	5	2	3
M4 Diárias	4	6	5	3	2	1
M4 Horárias	1	6	4	2	3	5
M4 Mensais	4	6	5	3	2	1
M4 Trimestrais	3	6	5	4	2	1
M4 Semanais	1	6	5	2	4	3
M4 Anuais	3	6	5	4	2	1
M5 Diárias	4	3	1	2	5	6
COVID-19 *Confirmed US*	3	6	1	2	4	5
COVID-19 *Deaths US*	4	6	1	2	3	5
COVID-19 *Confirmed Global*	3	6	2	4	1	5
COVID-19 *Deaths Global*	2	6	4	1	3	5
COVID-19 *Recovered Global*	4	6	2	1	3	5
TSDL Frequência 1	5	6	4	3	1	2
TSDL Frequência 4	2	6	5	3	1	4
TSDL Frequência 12	5	6	3	2	1	4
SOMA TOTAL	61	111	65	50	45	67
RANKING GERAL	3	6	4	2	1	5

Fonte: Autoria própria

Tabela 35 – *Ranking* de técnicas de seleção estabelecido com base nos valores de sMAPE de teste obtidos. Horizontes de predição originais.

Posição no *ranking*	Técnica de seleção/Combinação
1	Seleção NRegs + Fusão
2	Seleção 40p
3	auto.arima
4	Seleção NRegs
5	Proporção 80/20 (sem seleção de algoritmo)
6	AdaGrad

Fonte: Autoria própria

A análise de desempenho dos diferentes métodos utilizados nos experimentos foi realizada conforme proposto por Demšar (2006) para a comparação de classificadores. Para isso, foram utilizados os valores complementares de sMAPE obtidos (ou seja $2 - SMAPE_{calculado}$) de forma a garantir que o *ranking* da análise estatística fosse feito com

base no método que apresentasse menor erro.

Ao realizar um teste não-paramétrico de Friedman (FRIEDMAN, 1937) para o conjunto completo dos valores de sMAPE, obteve-se o seguinte:

```
Friedman's rank sum test
data: dsSMAPEsDatasets[, 2:7]
Friedman's chi-squared = 41.316, df = 5, p-value = 8.101e-08
```

O *valor de p* (*p-value = 8.101e-18*) obtido com o teste de Friedman indica que os valores de sMAPE obtidos pelos diferentes métodos para os diferentes *datasets* são estatisticamente diferentes entre si.

Aplicando um teste pareado de Nemenyi (*Nemenyi post-hoc test*) (NEMENYI, 1962), obtém-se os valores indicados na Figura 27 e o gráfico de diferença crítica da Figura 28.

Figura 27 – Dados obtidos com a aplicação de *Nemenyi post-hoc test* com valores de sMAPE de testes. Horizontes de predição originais.

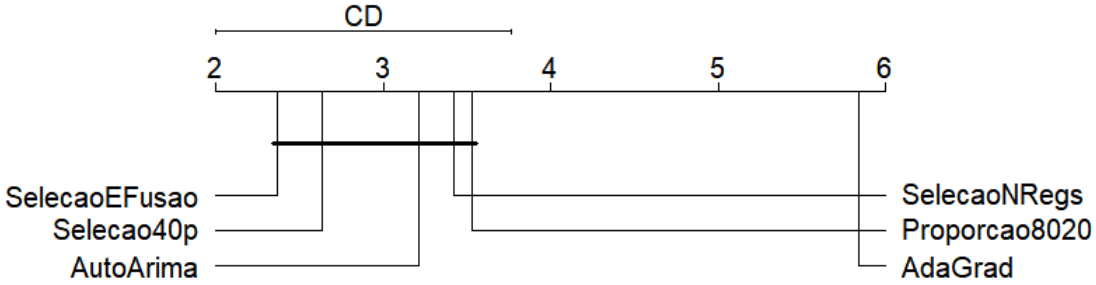
```
> test <- nemenyiTest (dsSMAPEsDatasets[,2:7], alpha=0.05)
> test$diff.matrix
      AutoArima   AdaGrad SelecaoNRegs Selecao40p SelecaoEFusao Proporcao8020
[1,]  0.0000000 -2.631579   -0.2105263  0.5789474     0.8421053    -0.3157895
[2,] -2.6315789  0.000000    2.4210526  3.2105263     3.4736842     2.3157895
[3,] -0.2105263  2.421053    0.0000000  0.7894737     1.0526316    -0.1052632
[4,]  0.5789474  3.210526    0.7894737  0.0000000     0.2631579    -0.8947368
[5,]  0.8421053  3.473684    1.0526316  0.2631579     0.0000000    -1.1578947
[6,] -0.3157895  2.315789   -0.1052632 -0.8947368    -1.1578947     0.0000000

> abs(test$diff.matrix) > test$statistic
     AutoArima AdaGrad SelecaoNRegs Selecao40p SelecaoEFusao Proporcao8020
[1,]     FALSE    TRUE        FALSE      FALSE         FALSE         FALSE
[2,]      TRUE   FALSE         TRUE       TRUE          TRUE          TRUE
[3,]     FALSE    TRUE        FALSE      FALSE         FALSE         FALSE
[4,]     FALSE    TRUE        FALSE      FALSE         FALSE         FALSE
[5,]     FALSE    TRUE        FALSE      FALSE         FALSE         FALSE
[6,]     FALSE    TRUE        FALSE      FALSE         FALSE         FALSE
```

Fonte: Autoria própria

É possível notar no gráfico de diferença crítica que o método que mais se distancia dos demais é o AdaGrad utilizado isoladamente. Isso ocorre no gráfico porque os valores de sMAPE de AdaGrad são expressivamente maiores que os apresentados pelos demais. No entanto, nessa análise não é possível observar diferença importante entre os demais métodos.

Figura 28 – Gráfico de diferença crítica (*CD plot*) com comparação de desempenho de diferentes técnicas de seleção. Horizontes de predição originais.



Fonte: Autoria própria

Tomando como base que o objetivo desta análise é verificar qual método combinado (auto.arima + AdaGrad) que mais se destaca, diferentes análises complementares foram efetuadas, a fim de identificar, principalmente, qual dos métodos avaliados pode apresentar melhores resultados que o uso isolado de auto.arima ou AdaGrad.

Tendo em vista a importância de se ter um método mais viável para a implementação com algoritmos de mineração de fluxos de dados, a técnica baseada na seleção de algoritmo calculada com base nos 40% dos registros finais de treinamento (Seleção40p) será desconsiderada nos resultados apresentados a seguir. O uso dessa estratégia pode ser viável em séries de curta duração; no entanto, fica inviável para os casos em que as séries venham a apresentar grande quantidade de observações. Ainda em relação à questão de uso futuro do método selecionado em cenários de fluxos de dados, um questionamento sobre o uso de auto.arima pode ser levantado, especialmente por se tratar de um método *batch*. No entanto, estudos futuros podem fazer a substituição do uso de auto.arima (utilizado nos experimentos deste estudo) por uma implementação de ARIMA com janelas deslizantes, viabilizando dessa forma a manutenção desse algoritmo na arquitetura da solução final.

Retirando AdaGrad e Seleção40p da análise, os valores do teste de Friedman passam a ser os seguintes:

```
Friedman's rank sum test
data: datasetTESTE
Friedman's chi-squared = 5.9684, df = 3, p-value = 0.1132
```

O valor de *p-value = 0.1132* indica que não foi observada diferença estatística significativa entre os valores. Uma análise com Nemenyi post-hoc test também não evidencia um método específico (Figura 29).

Figura 29 – Dados obtidos com a aplicação de *Nemenyi post-hoc test* com valores de sMAPE de testes, sem valores para AdaGrad nem Seleção40p. Horizontes de predição originais.

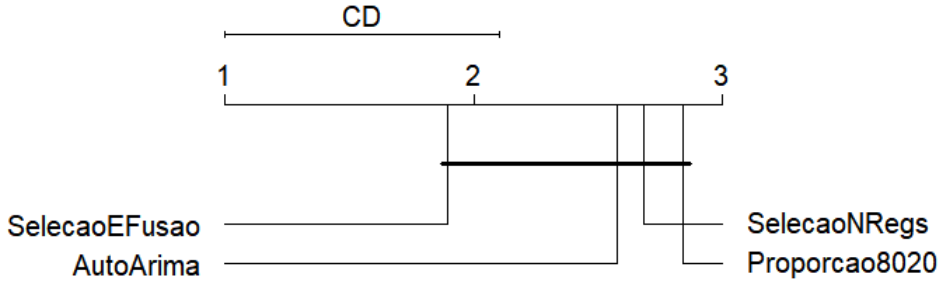
```
> test$diff.matrix
      AutoArima SelecaoNRegs SelecaoEFusao Proporcao8020
[1,]  0.0000000   -0.1052632     0.6842105    -0.2631579
[2,] -0.1052632    0.0000000     0.7894737    -0.1578947
[3,]  0.6842105    0.7894737     0.0000000    -0.9473684
[4,] -0.2631579   -0.1578947    -0.9473684     0.0000000

> abs(test$diff.matrix) > test$statistic
     AutoArima SelecaoNRegs SelecaoEFusao Proporcao8020
[1,]     FALSE        FALSE         FALSE         FALSE
[2,]     FALSE        FALSE         FALSE         FALSE
[3,]     FALSE        FALSE         FALSE         FALSE
[4,]     FALSE        FALSE         FALSE         FALSE
```

Fonte: Autoria própria

No entanto, fazendo a geração de um gráfico de diferença crítica (Figura 30) é possível observar que a técnica SeleçãoEFusão aparece em primeira posição no gráfico, sugerindo que se distancia dos demais métodos pelos valores que apresenta no *ranking*.

Figura 30 – Gráfico de diferença crítica (CD plot) com comparação de desempenho de diferentes técnicas de seleção, sem AdaGrad e sem Seleção com base em 40% dos registros. Horizontes de predição originais.



Fonte: Autoria própria

Apesar de os desempenhos não serem considerados estatisticamente diferentes (para nível de significância alpha = 0.05), é possível observar que o método SeleçãoEFusao é o que possui uma melhor posição no *ranking* em comparação aos demais métodos, o que o faz, para o objeto deste estudo, a técnica proposta para o método.

4.5.2 Análise de Ganhos Considerando Horizonte de Predição h=6

Os experimentos originais foram realizados para todos os *datasets* utilizando diferentes horizontes de predição, variando de acordo com a periodicidade das séries. Os valores de h (horizonte de predição) variaram de 6 *forecasts* (para séries anuais) a 48 *forecasts* (para séries horárias). Com o objetivo de trabalhar com valores de sMAPE para um horizonte de predição padrão, optou-se por uma análise limitada a *h=6*.

Uma análise de ganhos foi feita para as predições feitas para um horizonte de 6 predições e os valores obtidos podem ser observados na Tabela 36.

Tabela 36 – Tabela de ganhos percentuais obtidos em relação a auto.arima pelo uso de diferentes técnicas de seleção e/ou combinação. Calculados para horizonte de predição h=6.

Dataset	Tipo Série	Qtd	NRegs	40p	Fusão	%(80/20)	%(ARI/ADG)
M3	Mensais	1428	3,16%	2,90%	3,19%	3,01%	3,741% (60/40)
M3	Anuais	645	-0,99%	-0,73%	-0,31%	-3,28%	-0,271% (90/10)
M3	Trimestrais	756	0,22%	1,08%	0,66%	-4,74%	-1,345% (90/10)
M3	Outras	174	2,05%	-0,32%	1,58%	-1,47%	-0,446% (90/10)
M4	Diárias	4227	0,10%	0,14%	1,01%	0,29%	0,659% (90/10)
M4	Horárias	414	-3,42%	-0,73%	-0,28%	-3,13%	5,998% (90/10)
M4	Mensais	48000	0,37%	0,58%	0,83%	2,14%	2,163% (70/30)
M4	Trimestrais	24000	-1,01%	-0,06%	0,45%	0,85%	0,910% (90/10)
M4	Semanais	359	-9,20%	-0,21%	-2,73%	-2,08%	-0,515% (90/10)
M4	Anuais	23000	-0,73%	-0,10%	0,45%	1,03%	1,364% (90/10)
M5	Diárias	30490	13,47%	12,84%	-0,46%	-0,79%	-0,724% (90/10)
COVID-19	*Confirmed US*	3340	7,44%	7,46%	-0,42%	-16,72%	-15,310% (90/10)
COVID-19	*Deaths US*	3340	9,53%	8,60%	-0,07%	-6,23%	-5,092% (90/10)
COVID-19	*Confirmed Global*	266	0,37%	-1,55%	0,35%	-21,04%	-8,194% (90/10)
COVID-19	*Deaths Global*	266	-0,04%	0,68%	0,12%	-14,95%	-6,513% (90/10)
COVID-19	*Recovered Global*	253	2,73%	9,15%	1,53%	-68,81%	-65,619% (90/10)
TSDL	Frequência 1	299	1,08%	2,31%	5,53%	2,73%	2,731% (80/20)
TSDL	Frequência 4	63	-3,08%	-2,53%	15,11%	-3,44%	-1,180% (90/10)
TSDL	Frequência 12	238	6,81%	6,88%	9,27%	2,71%	3,530% (60/40)
	GANHO MÉDIO:		1,52%	2,44%	1,88%	-7,05%	

Fonte: Autoria própria

De maneira análoga ao que foi feito para os testes com horizontes de predição originais, uma vez analisados os ganhos observados pelas diferentes técnicas, uma análise de *ranking* baseada nos valores de sMAPE de teste foi realizada. Para isso, os valores de sMAPE de teste foram tabulados (Tabela 37) e utilizados para o estabelecimento de um *ranking* por *dataset* com base nos valores de sMAPE obtidos por cada método para horizonte de predição *h=6* (Tabela 38).

Tabela 37 – Tabela de sMAPE de teste obtidos pelo uso de diferentes técnicas de seleção e/ou combinação. Valores de sMAPE calculados para todas as séries de cada *dataset* e horizonte de predição h=6.

Dataset	auto.arima	AdaGrad	NRegs	40p	Fusão	%(80/20)
M3 Mensais	0,12340	0,12928	0,11950	0,11982	0,11946	0,11968
M3 Anuais	0,17104	0,32147	0,17273	0,17229	0,17157	0,17666
M3 Trimestrais	0,08906	0,14237	0,08887	0,08810	0,08848	0,09328
M3 Outras	0,03882	0,04975	0,03802	0,03894	0,03820	0,03939
M4 Diárias	0,02022	0,02615	0,02020	0,02020	0,02002	0,02017
M4 Horárias	0,14229	0,34272	0,14716	0,14333	0,14268	0,14675
M4 Mensais	0,09600	0,10705	0,09565	0,09545	0,09521	0,09395
M4 Trimestrais	0,09391	0,11253	0,09485	0,09396	0,09348	0,09311
M4 Semanais	0,06813	0,09497	0,07440	0,06827	0,06999	0,06955
M4 Anuais	0,15156	0,21407	0,15267	0,15171	0,15088	0,15000
M5 Diárias	1,37112	1,20903	1,18645	1,19511	1,37744	1,38189
COVID-19 *Confirmed US*	0,03978	0,05620	0,03682	0,03682	0,03995	0,04643
COVID-19 *Deaths US*	0,07825	0,09100	0,07080	0,07153	0,07831	0,08313
COVID-19 *Confirmed Global*	0,01349	0,03966	0,01344	0,01370	0,01344	0,01633
COVID-19 *Deaths Global*	0,02177	0,04471	0,02178	0,02163	0,02175	0,02503
COVID-19 *Recovered Global*	0,02405	0,05533	0,02340	0,02185	0,02368	0,04060
TSDL Frequência 1	0,34071	0,36697	0,33704	0,33283	0,32188	0,33141
TSDL Frequência 4	0,21719	0,27022	0,22387	0,22270	0,18438	0,22467
TSDL Frequência 12	0,40484	0,41533	0,37729	0,37699	0,36732	0,39389

Fonte: Autoria própria

Analisando os valores de *ranking* é possível observar que o método com SeleçãoE-Fusão aparece novamente como primeiro colocado no *ranking* para os sMAPEs calculados para *h=6*, sugerindo que o método é capaz de apresentar resultados com ganhos superiores aos observados com o uso isolado de auto.arima. A Tabela 39 apresenta o resultado da análise comparativa de *ranking* de sMAPEs de teste para *h=6*.

Aplicando o teste de Friedman sobre os *datasets* processados com *h=6* e excluindo da análise os valores de AdaGrad e Seleção40p, obtém-se os valores indicados abaixo:

```
Friedman's rank sum test
data: datasetTESTE
Friedman's chi-squared = 9.2368, df = 3, p-value = 0.0263
```

O valor de *p-value = 0.0263* indica (para nível de significância *alpha = 0.05*) que os valores obtidos são estatisticamente diferentes entre si. Aplicando um Nemenyi *post-hoc test* obtém-se os valores indicados na Figura 31.

Tabela 38 – Tabela de *ranking* de valores de sMAPE de teste obtidos pelo uso de diferentes técnicas de seleção e/ou combinação. Valores de sMAPE calculados para horizonte de predição h=6. *Ranking* estabelecido por *dataset*.

Dataset	auto.arima	AdaGrad	NRegs	40p	Fusão	%(80/20)
M3 Mensais	5	6	2	4	1	3
M3 Anuais	1	6	4	3	2	5
M3 Trimestrais	4	6	3	1	2	5
M3 Outras	3	6	1	4	2	5
M4 Diárias	5	6	3	3	1	2
M4 Horárias	1	6	5	3	2	4
M4 Mensais	5	6	4	3	2	1
M4 Trimestrais	4	6	5	3	2	1
M4 Semanais	1	6	5	2	4	3
M4 Anuais	3	6	5	4	2	1
M5 Diárias	4	3	1	2	5	6
COVID-19 *Confirmed US*	3	6	1	1	4	5
COVID-19 *Deaths US*	3	6	1	2	4	5
COVID-19 *Confirmed Global*	3	6	1	4	1	5
COVID-19 *Deaths Global*	3	6	4	1	2	5
COVID-19 *Recovered Global*	4	6	2	1	3	5
TSDL Frequência 1	5	6	4	3	1	2
TSDL Frequência 4	2	6	4	3	1	5
TSDL Frequência 12	5	6	3	2	1	4
SOMA TOTAL	64	111	58	49	42	72
RANKING GERAL	3	6	4	2	1	5

Fonte: Autoria própria

Tabela 39 – Ranking de técnicas de seleção estabelecido com base nos valores de sMAPE de teste obtidos. Horizonte de predição h=6.

Posição no *ranking*	Técnica de seleção/Combinação
1	Seleção NRegs + Fusão
2	Seleção 40p
3	auto.arima
4	Seleção NRegs
5	Proporção 80/20 (sem seleção de algoritmo)
6	AdaGrad

Fonte: Autoria própria

Nos testes com h=6 para todos os *datasets*, os resultados do teste de Friedman apresentam diferença estatística dentro do nível de significância estabelecido (*alpha=0.05*), indicando que os métodos avaliados possuem desempenho estatisticamente diferentes entre si.

Figura 31 – Dados obtidos com a aplicação de *Nemenyi post-hoc test* com valores de sMAPE de testes, sem valores para AdaGrad nem Seleção40p. Horizontes de predição h=6.

```
> test <- nemenyiTest (datasetTESTE, alpha=0.05)
> test$diff.matrix
      AutoArima SelecaoNRegs SelecaoEFusao Proporcao8020
[1,]  0.0000000    0.2368421     0.8684211    -0.3684211
[2,]  0.2368421    0.0000000     0.6315789    -0.6052632
[3,]  0.8684211    0.6315789     0.0000000    -1.2368421
[4,] -0.3684211   -0.6052632    -1.2368421     0.0000000

> abs(test$diff.matrix) > test$statistic
     AutoArima SelecaoNRegs SelecaoEFusao Proporcao8020
[1,]     FALSE        FALSE         FALSE         FALSE
[2,]     FALSE        FALSE         FALSE         FALSE
[3,]     FALSE        FALSE         FALSE          TRUE
[4,]     FALSE        FALSE          TRUE         FALSE
```

Fonte: Autoria própria

A Figura 32 apresenta o gráfico de diferença crítica para os valores de sMAPE de teste para h=6. É possível notar que o método de SeleçãoEFusão é um método que consegue se distanciar por sua posição no ranking em comparação com as posições de auto.arima e demais métodos avaliados. Isso sugere uma tendência que se repetiu em ambas as análises efetuadas até o momento (com h originais e h padronizado em h=6).

4.5.3 Análise de Ganhos Considerando Horizonte de Predição h=1

Uma análise similar às efetuadas para horizontes de predição originais e h=6 foi feita para horizonte de predição h=1 a fim de buscar uma diferença estatística mais significativa entre os resultados apresentados pelas diferentes técnicas.

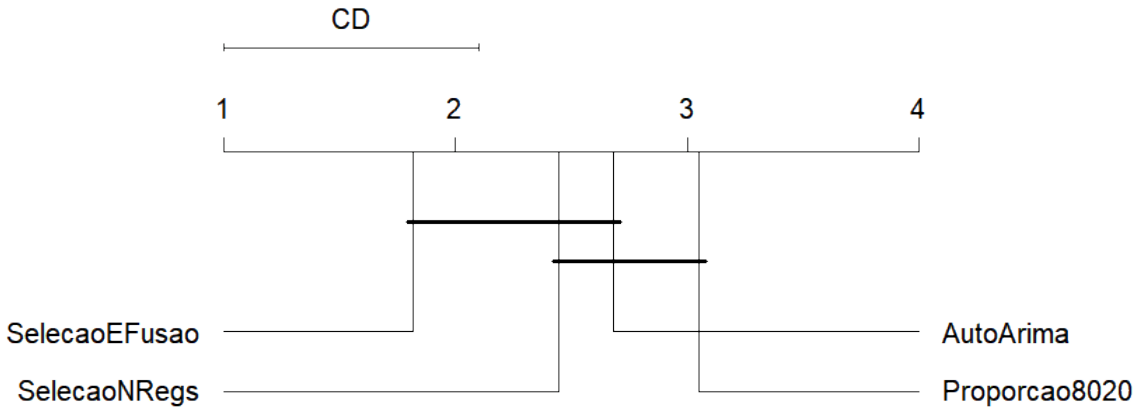
Uma análise de ganhos foi feita para as predições realizadas para um horizonte de h=1 (*one-step-ahead forecast*) e os valores obtidos podem ser observados na Tabela 40.

Para estabelecer um *ranking* de desempenho das diferentes técnicas avaliadas, uma tabela com os valores de sMAPE obtidos para h=1 foi montada (Tabela 41) e utilizada como base para a criação da tabela de *rankings* (Tabela 42).

De forma resumida, o *ranking* geral de desempenho das diferentes técnicas para h=1 pode ser observado na Tabela 43.

As tabelas de *ranking* evidenciam, novamente, que o método SeleçãoEFusão con-

Figura 32 – Gráfico de diferença crítica (*CD plot*) com comparação de desempenho de diferentes técnicas de seleção. Horizonte de predição h=6.



Fonte: Autoria própria

segue se manter como melhor colocado na comparação com os demais métodos quando se faz a contagem de vezes em que apresentou menor valor de sMAPE de teste quando comparado aos demais.

Aplicando o teste de Friedman sobre os *datasets* processados com h=1 e excluindo da análise os valores de AdaGrad e Seleção40p, obtém-se os valores indicados abaixo:

```
Friedman's rank sum test
data: datasetTESTE
Friedman's chi-squared = 22.374, df = 3, p-value = 5.454e-05
```

O valor de *p-value = 5.454e-05* indica (para nível de significância *alpha = 0.05*) que os valores obtidos são estatisticamente diferentes entre si. Aplicando um Nemenyi *post-hoc test* obtém-se os valores indicados na Figura 33.

A Figura 34 apresenta o gráfico de diferença crítica para os valores de sMAPE de teste para h=1 com o posicionamento dos diferentes métodos. É possível notar que o método de SeleçãoEFusão é um método que consegue se distanciar por sua posição no *ranking* em comparação com as posições de auto.arima e demais métodos avaliados.

Na próxima subseção os resultados das análises feitas para horizontes de predição originais, h=6 e h=1 são analisados em conjunto de forma a verificar o desempenho da técnica SeleçãoEFusão (que corresponde à técnica utilizada pela configuração *default* do método AA-ACF proposto por este estudo) em comparação às demais. Na sequência, é apresentada a discussão final sobre os resultados.

Tabela 40 – Tabela de ganhos percentuais obtidos em relação a auto.arima pelo uso de diferentes técnicas de seleção e/ou combinação. Calculados para horizonte de predição h=1 (*one-step-ahead forecast*).

Dataset	Tipo Série	Qtd	NRegs	40p	Fusão	%(80/20)	%(ARI/ADG)
M3	Mensais	1428	1,010%	1,288%	1,502%	0,480%	0,519% (90/10)
M3	Anuais	645	0,755%	1,643%	0,523%	-12,073%	-3,925% (90/10)
M3	Trimestrais	756	-0,304%	0,841%	0,536%	-10,208%	-4,016% (90/10)
M3	Outras	174	-1,797%	-0,547%	-0,658%	-6,064%	-2,154% (90/10)
M4	Diárias	4227	-1,027%	0,067%	0,214%	-2,964%	-0,686% (90/10)
M4	Horárias	414	-0,980%	0,911%	4,203%	18,188%	18,188% (90/20)
M4	Mensais	48000	0,006%	0,706%	0,762%	0,557%	0,854% (90/10)
M4	Trimestrais	24000	-1,174%	0,036%	0,387%	-0,375%	0,442% (90/10)
M4	Semanais	359	-26,694%	-2,107%	-12,132%	-12,142%	-5,466% (90/10)
M4	Anuais	23000	-0,083%	0,029%	0,750%	-1,559%	0,331% (90/10)
M5	Diárias	30490	15,476%	14,957%	-0,376%	-0,757%	-0,700% (90/10)
COVID-19	*Confirmed US*	3340	10,998%	11,615%	-1,557%	-33,887%	-30,707% (90/10)
COVID-19	*Deaths US*	3340	15,328%	14,909%	-0,902%	-12,183%	-9,206% (90/10)
COVID-19	*Confirmed Global*	266	0,579%	-0,146%	0,796%	-44,003%	-17,384% (90/10)
COVID-19	*Deaths Global*	266	-2,558%	0,342%	-1,310%	-24,847%	-10,426% (90/10)
COVID-19	*Recovered Global*	253	5,143%	-19,591%	2,681%	-158,055%	-149,706% (90/10)
TSDL	Frequência 1	299	-0,927%	-0,202%	6,218%	-1,102%	-0,378% (90/10)
TSDL	Frequência 4	63	-3,991%	-3,768%	32,726%	-5,493%	-3,293% (90/10)
TSDL	Frequência 12	238	0,678%	-2,079%	5,177%	-1,243%	-0,480% (90/10)
	GANHO MÉDIO:		0,549%	0,995%	2,081%	-16,196%	

Fonte: Autoria própria

4.5.4 Análises complementares

Na seção anterior foram efetuadas várias análises sobre os ganhos obtidos pelo uso do método em diferentes horizontes de predição. Nesta seção algumas análises complementares são efetuadas com o objetivo de: avaliar os ganhos obtidos em diferentes horizontes de predição; analisar a correlação entre os ganhos obtidos e as características inerentes às séries; e verificar os valores do oráculo para identificar potencial de melhoria no método proposto pelo estudo.

Observando os valores do quadro comparativo de ganhos na Tabela 44, nota-se que a posição das diferentes técnicas no *ranking* não se altera ao fazer a análise do erro para diferentes horizontes de predição. Numa comparação dos dois principais métodos em análise (SeleçãoNRegs e SeleçãoEFusão), os valores mostram que o maior ganho é obtido com o uso do método que aparece em 1^0 lugar no *ranking*, no caso SeleçãoEFusão. Apesar de o método de Seleção40p apresentar valores médios de ganho maiores, esse método (viável para processos *batch*, com séries de dimensões previamente conhecidas) é de aplicação limitada em cenários de fluxos de dados.

Tabela 41 – Tabela de sMAPE de teste obtidos pelo uso de diferentes técnicas de seleção e/ou combinação. Valores de sMAPE calculados para todas as séries de cada *dataset* e horizonte de predição h=1 (*one-step-ahead forecast*).

Dataset	auto.arima	AdaGrad	NRegs	40p	Fusão	%(80/20)
M3 Mensais	0,11364	0,12839	0,11250	0,11218	0,11194	0,11310
M3 Anuais	0,08233	0,20069	0,08171	0,08098	0,08190	0,09227
M3 Trimestrais	0,05185	0,09740	0,05201	0,05142	0,05158	0,05715
M3 Outras	0,01741	0,02809	0,01772	0,01751	0,01753	0,01847
M4 Diárias	0,01014	0,01529	0,01024	0,01013	0,01012	0,01044
M4 Horárias	0,15750	0,26591	0,15904	0,15607	0,15088	0,12885
M4 Mensais	0,06625	0,08473	0,06625	0,06578	0,06574	0,06588
M4 Trimestrais	0,05991	0,07921	0,06061	0,05989	0,05968	0,06013
M4 Semanais	0,04284	0,08181	0,05428	0,04375	0,04804	0,04805
M4 Anuais	0,08288	0,13137	0,08295	0,08285	0,08226	0,08417
M5 Diárias	1,36513	1,17270	1,15387	1,16096	1,37027	1,37547
COVID-19 *Confirmed US*	0,02221	0,03862	0,01977	0,01963	0,02255	0,02973
COVID-19 *Deaths US*	0,04914	0,06466	0,04161	0,04182	0,04959	0,05513
COVID-19 *Confirmed Global*	0,00584	0,02587	0,00580	0,00584	0,00579	0,00840
COVID-19 *Deaths Global*	0,01225	0,03210	0,01256	0,01221	0,01241	0,01529
COVID-19 *Recovered Global*	0,01075	0,04169	0,01020	0,01285	0,01046	0,02774
TSDL Frequência 1	0,27698	0,32132	0,27954	0,27754	0,25975	0,28003
TSDL Frequência 4	0,16600	0,20619	0,17263	0,17226	0,11168	0,17512
TSDL Frequência 12	0,32688	0,39494	0,32466	0,33368	0,30996	0,33094

Fonte: Autoria própria

Uma análise adicional foi efetuada na tentativa de identificar a correlação entre as características inerentes das séries e os ganhos obtidos para diferentes horizontes de predição pelo método SeleçãoEFusão. Com o uso de funções do pacote *tsfeatures* (HYNDMAN et al., 2019), foram extraídas características das séries de todos os *datasets* do experimento.

Uma planilha auxiliar foi montada com as seguintes informações: nome do dataset; ganho médio obtido com o método SeleçãoEFusão para horizontes de predição originais, h=6 e h=1; valores médios das *features* extraídas com o pacote *tsfeatures* incluindo *length, trend, spike, linearity, curvature, e_acf1, e_acf10, entropy, x_acf1, x_acf10, diff1_acf1, diff1_acf10, diff2_acf1, diff2_acf10, stability, lumpiness, max_level_shift, time_level_shift, max_var_shift, time_var_shift, max_kl_shift, time_kl_shift, crossing_points, flat_spots, hurst, unitroot_kpss, unitroot_pp, heterogeneity_arch_acf, heterogeneity_garch_acf, heterogeneity_arch_r2, heterogeneity_garch_r2* e *nonlinearity*. As *features* computadas e consideradas no estudo, pertencentes ao pacote *tsfeatures*, estão descritas no endereço https://cran.r-project.org/web/packages/tsfeatures/vignettes/tsfeatures.html. As *features* mais relevantes estão descritas mais adiante nesta seção.

Os valores médios de tais *features* foram comparados com os ganhos obtidos para

Tabela 42 – Tabela de *ranking* de valores de sMAPE de teste obtidos pelo uso de diferentes técnicas de seleção e/ou combinação. Valores de sMAPE calculados para horizonte de predição h=1 (*one-step-ahead forecast*). *Ranking* estabelecido por *dataset*.

Dataset	auto.arima	AdaGrad	NRegs	40p	Fusão	%(80/20)
M3 Mensais	5	6	3	2	1	4
M3 Anuais	4	6	2	1	3	5
M3 Trimestrais	3	6	4	1	2	5
M3 Outras	1	6	4	2	3	5
M4 Diárias	3	6	4	2	1	5
M4 Horárias	4	6	5	3	2	1
M4 Mensais	4	6	4	2	1	3
M4 Trimestrais	3	6	5	2	1	4
M4 Semanais	1	6	5	2	3	4
M4 Anuais	3	6	4	2	1	5
M5 Diárias	4	3	1	2	5	6
COVID-19 *Confirmed US*	3	6	2	1	4	5
COVID-19 *Deaths US*	3	6	1	2	4	5
COVID-19 *Confirmed Global*	3	6	2	3	1	5
COVID-19 *Deaths Global*	2	6	4	1	3	5
COVID-19 *Recovered Global*	3	6	1	4	2	5
TSDL Frequência 1	2	6	4	3	1	5
TSDL Frequência 4	2	6	4	3	1	5
TSDL Frequência 12	3	6	2	5	1	4
SOMA TOTAL:	56	111	61	43	40	86
RANKING GERAL:	3	6	4	2	1	5

Fonte: Autoria própria

Tabela 43 – Ranking de técnicas de seleção estabelecido com base nos valores de sMAPE de teste obtidos. Horizonte de predição h=1 (*one-step-ahead forecast*).

Posição no *ranking*	Técnica de seleção/Combinação
1	Seleção NRegs + Fusão
2	Seleção 40p
3	auto.arima
4	Seleção NRegs
5	Proporção 80/20 (sem seleção de algoritmo)
6	AdaGrad

Fonte: Autoria própria

os diferentes horizontes e uma matriz de correlação de *Pearson* foi montada a fim de identificar as características das séries que mais se correlacionam com os resultados.

A Figura 35 mostra a matriz de correlação completa e a Figura 36 apresenta um

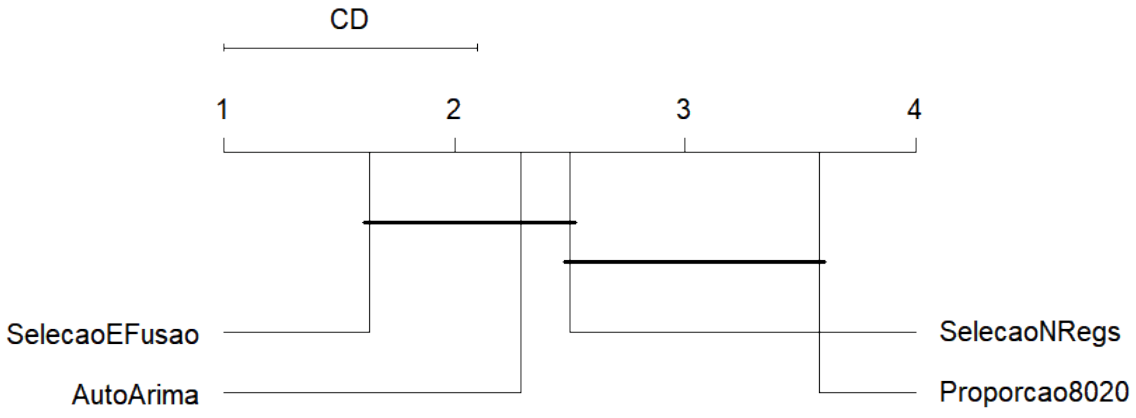
Figura 33 – Dados obtidos com a aplicação de *Nemenyi post-hoc test* com valores de sMAPE de testes, sem valores para AdaGrad nem Seleção40p. Horizontes de predição h=1 (*one-step-ahead forecast*).

```
> test <- nemenyiTest (datasetTESTE, alpha=0.05)
> test$diff.matrix
      AutoArima SelecaoNRegs SelecaoEFusao Proporcao8020
[1,]  0.0000000    0.2368421     0.8684211    -0.3684211
[2,]  0.2368421    0.0000000     0.6315789    -0.6052632
[3,]  0.8684211    0.6315789     0.0000000    -1.2368421
[4,] -0.3684211   -0.6052632    -1.2368421     0.0000000

> abs(test$diff.matrix) > test$statistic
     AutoArima SelecaoNRegs SelecaoEFusao Proporcao8020
[1,]     FALSE        FALSE         FALSE         FALSE
[2,]     FALSE        FALSE         FALSE         FALSE
[3,]     FALSE        FALSE         FALSE          TRUE
[4,]     FALSE        FALSE          TRUE         FALSE
```

Fonte: Autoria própria

Figura 34 – Gráfico de diferença crítica (*CD plot)* com comparação de desempenho de diferentes técnicas de seleção. Horizonte de predição h=1 (*one-step-ahead forecast*).



Fonte: Autoria própria

detalhe com a indicação das características que mais se correlacionam com os ganhos.

Observando em detalhe a parte inferior da matriz de correlação, pode-se identificar que os atributos com maior correlação (positiva e negativa) com os ganhos médios são *diff2_acf10* e *e_acf1*.

Tabela 44 – *Ranking* de técnicas de seleção estabelecido com base nos valores de sMAPE de teste obtidos. Horizontes de predição originais, h=6 e h=1.

Horizontes de predição originais (h=6 a h=48)			**Horizontes de predição padronizados em h=6**			**Horizontes de predição padronizados em h=1**		
Posição	**Método**	**Ganho**	**Posição**	**Método**	**Ganho**	**Posição**	**Método**	**Ganho**
1	Seleção NRegs + Fusão	1,65%	1	Seleção NRegs + Fusão	1,88%	1	Seleção NRegs + Fusão	2,081%
2	Seleção 40p	1,84%	2	Seleção 40p	2,44%	2	Seleção 40p	0,995%
3	auto.arima	NA	3	auto.arima	NA	3	auto.arima	NA
4	Seleção NRegs	0,87%	4	Seleção NRegs	1,52%	4	Seleção NRegs	0,549%
5	Proporção 80/20	-4,13%	5	Proporção 80/20	-7,05%	5	Proporção 80/20	-16,196%
6	AdaGrad	NC	6	AdaGrad	NC	6	AdaGrad	NC

Legenda: NA=Não se aplica; NC=Não calculado

Fonte: Autoria própria

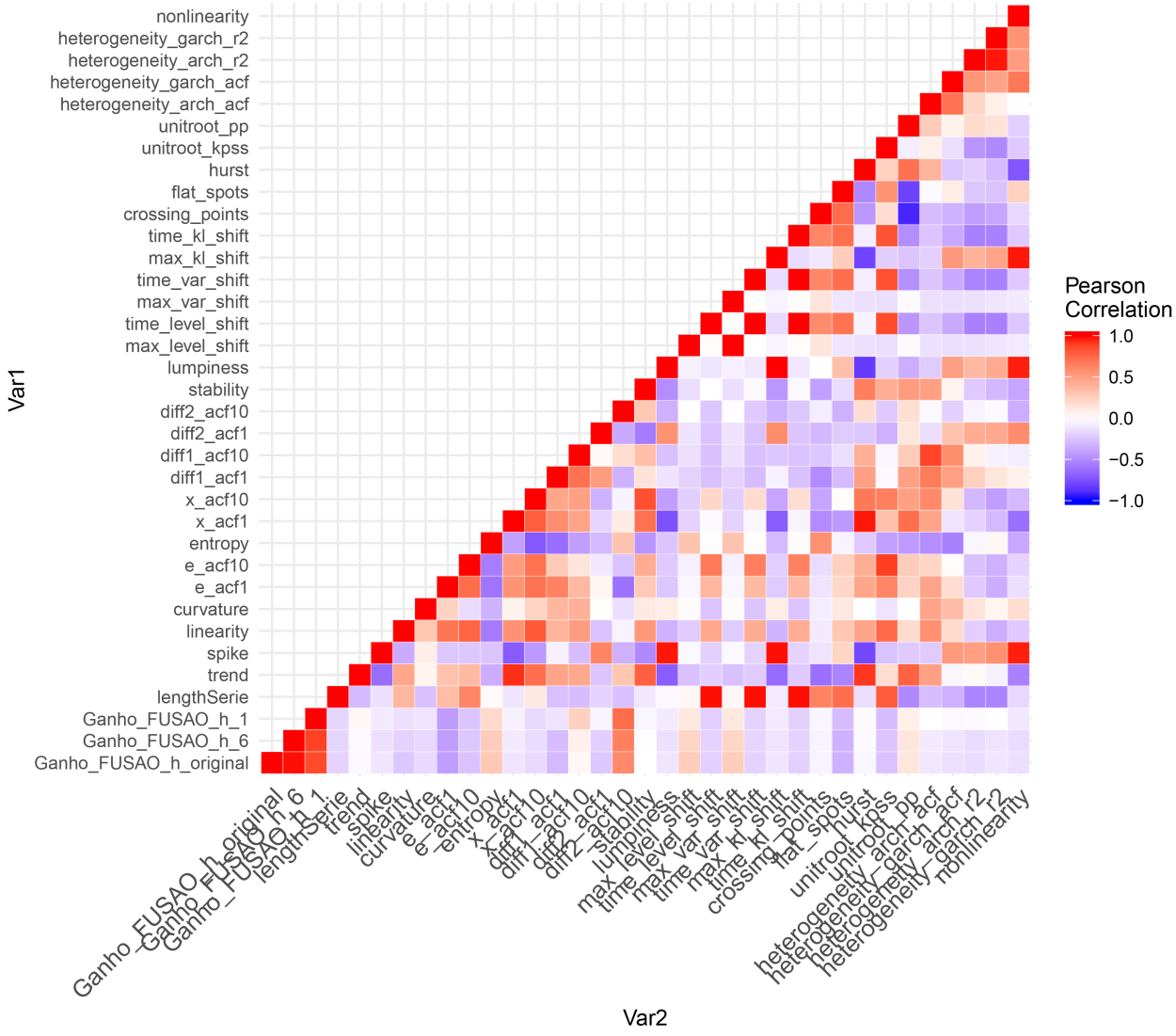
Tabela 45 – *Features* das séries extraídas com o pacote tsfeatures com maior correlação com os ganhos obtidos.

Ganho	Atributo	Correlação de Pearson
Ganho médio para h Originais	diff2_acf10	0,60
	entropy	0,29
	e_acf1	-0,42
	flat_spots	-0,33
Ganho médio para h = 6	diff2_acf10	0,65
	entropy	0,27
	e_acf1	-0,42
	flat_spots	-0,33
Ganho médio para h = 1	diff2_acf10	0,73
	entropy	0,25
	e_acf1	-0,43
	flat_spots	-0,29

Fonte: Autoria própria

Os valores da Tabela 45 mostram as quatro *features* que apresentaram maior correlação (duas positivas e duas negativas) em relação aos ganhos apresentados pelo método. A *feature* com maior correlação positiva com os ganhos médios obtidos com a técnica de SeleçãoEFusão é a ***diff2_acf10***, que corresponde à soma dos quadrados dos primeiros 10 coeficientes ACF obtidos da série que teve seus dados diferenciados duas vezes (*twice-differenced*).

Figura 35 – Matriz de correlação de *features* das séries analisadas versus ganhos obtidos pelas diferentes técnicas/métodos de seleção.

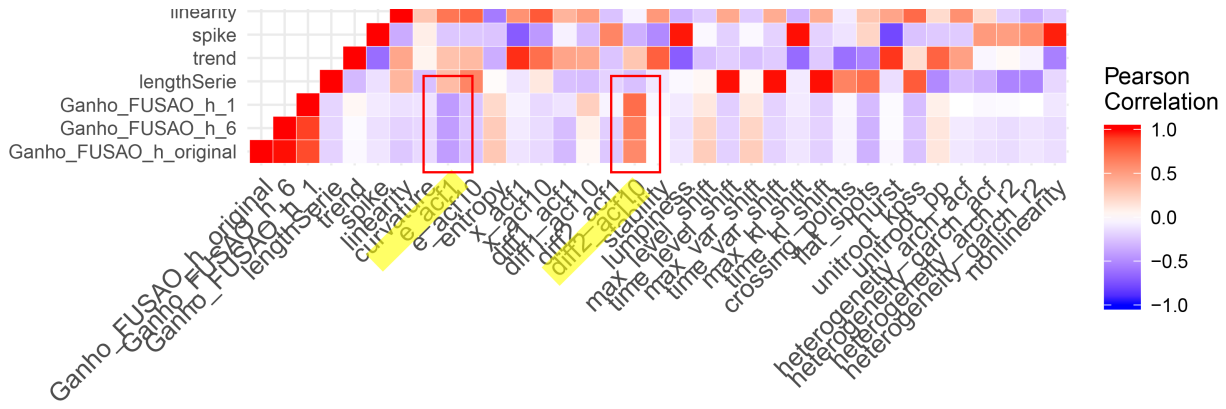


Fonte: Autoria própria

As outras características da tabela são descritas (de acordo com a documentação do pacote *tsfeatures*) como: ***entropy*** corresponde a entropia espectral de Shannon, uma medida de previsibilidade de uma série temporal, onde valores baixos indicam uma alta razão sinal-ruído e grandes valores ocorrem quando uma série é de difícil predição; ***e_acf1*** corresponde ao primeiro coeficiente de autocorrelação de e_t, onde e_t é o componente residual da decomposição STL de uma série temporal, que separa a série em seus componentes de sazonalidade, tendência e componente residual; e ***flat_spots*** são calculados dividindo o espaço de amostra de uma série temporal em dez intervalos de tamanhos iguais e computando a duração máxima de execução em qualquer intervalo único.

Essa informação pode ser útil para abordagens utilizadas em estudos futuros

Figura 36 – Detalhe da matriz de correlação de *features* das séries analisadas versus ganhos obtidos pelas diferentes técnicas/métodos de seleção.



Fonte: Autoria própria

do método com o objetivo de melhorar o processo de seleção de algoritmo levando em consideração o atributo *diff2_acf10*, visto que é o atributo que apresentou maior nível de correlação com os ganhos para todos os horizontes medidos.

Outra análise efetuada foi a verificação do Oráculo de AdaGrad e auto.arima, que são os principais algoritmos considerados no estudo. Para essa análise, os valores de sMAPE de teste do oráculo para cada *dataset* são calculados considerando o algoritmo que apresentou o menor sMAPE de teste para a série entre esses dois métodos usados isoladamente. Para efeito de comparação, os valores do Oráculo são apresentados na Tabela 46 em conjunto com os valores obtidos com auto.arima, AdaGrad e com o método de seleção proposto por este estudo, no caso o que utiliza a técnica de SeleçãoEFusão (indicada na tabela como Fusão).

Os valores do oráculo servem como um referencial do cenário ideal no qual os menores valores de erros de testes são obtidos para as séries. Os valores são teóricos pois seriam obtidos apenas se o método de seleção conseguisse antever em 100% dos casos o algoritmo original ideal para cada série em análise. No entanto, servem como um parâmetro a ser tomado como referência em busca de melhorias no processo de seleção.

Os cálculos para o oráculo foram feitos considerando os valores de predições obtidos pelo uso isolado de auto.arima e AdaGrad. Por esse motivo, seus valores serão menores que os apresentados pelos algoritmos que foram levados em consideração na análise. Isso pode ser percebido na tabela ao comparar os valores de sMAPE de teste apresentados pelo Oráculo, por auto.arima e por AdaGrad.

Os valores obtidos com a técnica de SeleçãoEFusão não entraram na definição do oráculo; no entanto, na Tabela 46 um dado interessante é o valor de sMAPE de teste

Tabela 46 – Tabela comparativa de valores do Oráculo e as principais técnicas do estudo.

		# Séries do Oráculo		sMAPE de Teste			
Dataset	# Séries	auto.arima	AdaGrad	Oráculo	auto.arima	AdaGrad	Fusão
M3 Mensais	1428	819	609	0,12521	0,14918	0,15586	0,1434
M3 Anuais	645	491	154	0,14752	0,17104	0,32147	0,17157
M3 Trimestrais	756	537	219	0,08644	0,10011	0,15652	0,09973
M3 Outras	174	116	58	0,03824	0,04513	0,05454	0,04419
M4 Diárias	4227	2702	1525	0,02660	0,03186	0,04063	0,03145
M4 Horárias	414	383	31	0,13554	0,1409	0,27325	0,14242
M4 Mensais	48000	30273	17727	0,11312	0,13495	0,14791	0,13427
M4 Trimestrais	24000	15360	8640	0,09006	0,1042	0,12386	0,1038
M4 Semanais	359	238	121	0,07448	0,08591	0,1144	0,08889
M4 Anuais	23000	14507	8493	0,12006	0,15156	0,21407	0,15089
M5 Diárias	30490	14435	16055	1,16556	1,36878	1,22966	1,37509
COVID Confirmed US	3340	2347	993	0,04839	0,05848	0,08011	0,05853
COVID Deaths US	3340	2484	856	0,09434	0,1122	0,12698	0,11211
COVID Confirmed Global	266	203	63	0,02407	0,02783	0,0676	0,02779
COVID Deaths Global	266	218	48	0,03352	0,03671	0,06557	0,03675
COVID Recovered Global	253	186	67	0,03502	0,04633	0,08349	0,0457
TSDL Frequência 1	299	190	109	0,30518	0,34071	0,36697	0,32188
TSDL Frequência 4	63	50	13	0,21221	0,22039	0,27841	**0,19299**
TSDL Frequência 12	238	107	131	0,38058	0,44038	0,45445	0,4038
Total de Séries	141558	85646	55912				

Fonte: Autoria própria

apresentado por essa técnica para o *dataset* TSDL Frequência 4 (sMAPE 0,19299), em que foi capaz de alcançar um valor menor que o estabelecido pelo oráculo de auto.arima e AdaGrad (sMAPE 0,21221). Isso foi possível pois a técnica de SeleçãoEFusão faz a seleção de algoritmo com base no erro de treinamento e, após isso, aplica uma fusão dos *forecasts* de AdaGrad e auto.arima. O cálculo do oráculo, por sua vez, leva em consideração os valores originais dos *forecasts* dos algoritmos estatístico e DSM, sem fazer nenhuma alteração ou fusão de valores.

Esse comportamento do valor de sMAPE da técnica de SeleçãoEFusão para TSDL Frequência 4 em relação aos valores do oráculo reforça a ideia de que a combinação de algoritmos é capaz de superar (em determinados cenários) os valores obtidos pelos algoritmos originais isolados. Estudos futuros devem ter como objetivo estender esse ganho da técnica de SeleçãoEFusão para outros *datasets*, tendo os valores do oráculo como referência para melhorias do método.

4.5.5 Discussão de Resultados

Ao analisar os resultados dos testes de *Friedman* aplicados para as análises de erros de testes obtidos para o processamento com horizontes de predição originais, *h=6* e *h=1* é possível observar diferenças estatísticas dentro do nível de significância estabelecido (*alpha=0.05*). Os gráficos de diferença crítica, no entanto, não demonstram um distanciamento entre as diferentes técnicas além da distância crítica calculada. Mas é importante destacar a tendência de melhores resultados apresentados pela técnica de SeleçãoEFusão que se evidenciou nos experimentos e pode ser observada nos gráficos.

Tanto o *ranking* de classificação dos métodos como os valores de ganhos percentuais em relação a auto.arima usado isoladamente, sugerem que a técnica de SeleçãoEFusão se mostra promissora. São observados, no entanto, *datasets* para os quais não foi possível alcançar ganhos positivos, chegando a serem percebidos ganhos negativos na faixa de -0,376% a -1,557% para 5 dos 19 datasets/subconjuntos considerados no experimento. Além disso, o *dataset* de séries semanais de M4 também não apresentou bons resultados com o uso da SeleçãoEFusão, tendo sido o que apresentou o maior valor de ganho negativo (-12,132%) com o uso desse método.

Devido aos ganhos negativos observados, deve ser material de estudo futuro investir em aprimoramento da técnica de *feature engineering* com o objetivo de melhorar os resultados para AdaGrad e, por sua vez, os valores finais obtidos com o *ensemble*.

Em contrapartida, ao analisar os resultados obtidos para o *dataset* TSDL, observa-se que foram possíveis atingir ganhos percentuais bem expressivos em comparação ao referencial auto.arima:

Para h = 6:

- Ganho de 5,53% para séries de frequência=1
- Ganho de 15,11% para séries de frequência=4
- Ganho de 9,27% para séries de frequência=12

Para h = 1:

- Ganho de 6,22% para séries de frequência=1
- Ganho de 32,73% para séries de frequência=4
- Ganho de 5,18% para séries de frequência=12

Analisando em conjunto os resultados obtidos com o processamento das 141.558 séries (correspondente à soma do total de séries de todos os *datasets* avaliados, M3, M4,

M5, COVID-19 e TSDL), é possível observar que a técnica de SeleçãoEFusão conseguiu atingir ganhos médios positivos (em relação ao uso isolado de auto.arima) para todos os horizontes de predição avaliados: ganho de 1,65% para *h=originais* (6 a 48 *forecasts*), ganho de 1,88% para *h=6* e ganho de 2,081% para *h=1*.

Os valores observados nos experimentos sugerem que o método baseado em SeleçãoEFusão tem a capacidade de se destacar das demais técnicas avaliadas. Isso fica evidente pelo método ter se posicionado como 1^0 lugar na contabilização (*ranking*) de número de ocorrências de menor valor de sMAPE de teste, bem como pelos ganhos médios apresentados.

Um fato curioso observado ao analisar as características obtidas com a função *tsfeatures()* é que a *feature* que mostrou maior correlação com os ganhos (a *feature diff2_acf10*) é obtida com base em coeficientes de autocorrelação da série. Isso sugere que a abordagem de utilizar coeficientes de autocorrelação como a base para o processo de *feature engineering* do método é acertada e merece aprofundamento dos estudos de forma a ampliar os ganhos obtidos. Pode-se avaliar, inclusive, o uso de *features* baseadas em ACF como critério de seleção de algoritmo, pois ao tornar o processo de seleção mais assertivo (de forma a indicar com maior nível de acerto as séries melhor processadas com AdaGrad), acredita-se que possa ser observado maior ganho no processo de predição como um todo.

Os autores em (BABCOCK et al., 2002) descrevem que, em um ambiente de *data streams*, o fluxo de dados chega de forma *online*; em alguns processos, inclusive, o sistema não tem controle sobre a ordem que os elementos de dados chegam para serem processados. Além disso, os fluxos de dados são potencialmente ilimitados em relação à quantidade de dados. Considerando isso, uma vez que um elemento do fluxo de dados foi processado, ele é descartado ou arquivado e não pode ser recuperado facilmente, a menos que seja armazenado (temporariamente) em memória (que geralmente é pequena em relação ao tamanho do fluxo).

O método AA-ACF faz uso combinado de algoritmos com técnicas distintas ao utilizar um algoritmo estatístico e um algoritmo de data stream mining. Devido ao modelo de implementação adotado em sua versão inicial, seu uso está limitado ao processamento batch de séries temporais, não oferecendo suporte ao processamento de fluxos contínuos de dados. Apesar disso, ao utilizar o algoritmo AdaGrad como um de seus componentes, foi possível verificar que um algoritmo adaptativo, que cria seu modelo de aprendizagem de forma incremental, é capaz de colaborar de forma positiva no alcance de melhores resultados ao ser utilizado de forma combinada com outra técnica. Com base no que foi observado pelos experimentos, um dos fatores que contribui para isso está relacionado ao fato de que, conforme a Tabela 46 (que apresenta o oráculo de séries de auto.arima e AdaGrad), do total de 141.558 séries utilizadas no experimento, o algoritmo AdaGrad apresentou melhores valores de predição para 55.912 séries (correspondendo a 39,5% do

total).

Em relação ao custo computacional da solução, é necessário observar que o método exige o treinamento com dois algoritmos distintos pelo fato de não se saber, de antemão, qual o método que apresentará melhores resultados no cálculo dos forecasts finais para as séries. Em relação à forma como está implementado atualmente, um ponto que merece adequações em trabalhos futuros é a diminuição de sua dependência de uso de arquivos externos, com o objetivo de diminuir operações de entrada e saída (E/S), que impactam no tempo de processamento do método.

Em resumo, com os resultados apresentados, pode-se entender que o método AA-ACF tem capacidade de alcançar resultados que o caracterizam como um método competitivo, com desempenho capaz de superar em diferentes cenários o método estatístico auto.arima utilizado isoladamente.

TESTE DE HIPÓTESE:

Os resultados obtidos pelo método AA-ACF, expressos pelos ganhos percentuais apresentados, demonstram que a hipótese estabelecida para o estudo (*Hipótese: "Combinação de algoritmos estatísticos com algoritmos de data stream mining possibilita melhor acurácia no processo de forecasting de multisséries temporais"*) foi confirmada. Isso sugere que o método é promissor e, em trabalhos futuros, deve passar por melhorias, em especial, no processo de seleção de algoritmos de forma a ampliar sua precisão na escolha do algoritmo mais apropriado para cada série.

5 Conclusão

A busca por técnicas inovadoras para resolver problemas clássicos é sempre um processo desafiador. A predição de demanda, mais especificamente a relacionada ao *forecasting* de séries temporais, por exemplo, é bem resolvida por técnicas que apresentam excelente desempenho, são fáceis de usar e possuem literatura abundante. No entanto, é na busca por diferentes técnicas, ou mesmo na combinação de diferentes linhas de pensamento, que se encontra a oportunidade de se criar algo novo, geralmente inspirado por uma nova maneira de pensar. As técnicas de mineração de fluxos de dados estão cada vez mais difundidas em cenários em que o volume de informações é crescente, como na área de redes sociais, nas aplicações *IoT*, nos ambientes que inspiram o conceito de *Big Data*.

Nesse tipo de cenário, em que é inviável considerar o uso de processos em lote para fazer a previsão de dados, ou em que a previsão de valores futuros deve ser em tempo real, existe uma demanda constante pela busca de novas tecnologias.

Espera-se que o método apresentado neste estudo possa servir de base para pesquisas futuras, uma vez que o universo de dados tem um comportamento dinâmico e demanda pesquisa constante para alcançar, cada vez mais, melhores respostas em tempo real, com maior precisão, sem dependência da disponibilidade total de dados históricos para previsões de séries temporais.

Os resultados obtidos pelos experimentos realizados com diferentes conjuntos de séries temporais mostram que o método AA-ACF é capaz de apresentar resultados com ganhos positivos em relação à acurácia apresentada pelo uso isolado dos algoritmos envolvidos, em especial em relação ao método auto.arima considerado como *benchmark* neste estudo.

A utilização de coeficientes de autocorrelação como base para o processo de *feature engineering* também se mostrou uma abordagem com potencial para auxiliar no processo de treinamento e predição. Um resultado interessante observado nos estudos foi o de perceber que os ganhos obtidos pelo método mostram correlação positiva com características extraídas das séries que também correspondem a dados obtidos a partir desses mesmos coeficientes. Isso reforça a percepção de que a abordagem é promissora e merece ser mais estudada.

Para este estudo, o método foi implementado em uma versão experimental capaz de desenvolver as análises necessárias para avaliar o comportamento dos algoritmos envolvidos, sem disponibilização de uma versão de distribuição (*deploy*). Nessa versão, os experimentos foram realizados com o processamento *batch* de séries temporais estáticas, de comprimento previamente conhecido, numa tentativa de avaliar o comportamento/ desempenho de

algoritmos de mineração de fluxos de dados em séries de duração relativamente curta, que são comumente encontradas em problemas de análise de predição de demanda como as relacionados a processos de compra e venda de produtos, ou em análises de dados sócio-econômicos (como censos populacionais), entre outros.

Utilizando essa versão, foi possível analisar a capacidade de AdaGrad em desenvolver seu modelo de aprendizagem e observar seu potencial para alcançar resultados competitivos. Uma demonstração disso pôde ser observada na análise das séries atribuídas a ele pelo oráculo, que demonstram que, para os *datasets* avaliados, ele tem o potencial de obter melhores predições em 39,5% das séries. Trabalhos futuros, então, devem incluir a melhoria no processo de seleção do método AA-ACF de forma a ampliar o número de séries direcionadas para predição com o algoritmo AdaGrad. Para isso, recomenda-se estudar de forma mais aprofundada diferentes estratégias para utilizar características inerentes das séries temporais no processo de seleção, a fim de tornar a escolha de algoritmo de predição mais assertiva e os erros de predição menores e mais competitivos.

Além disso, é preciso adaptar a estrutura interna do método de forma a realizar o processamento de fluxos de dados contínuos, a fim de caracterizá-lo como um método adaptativo, procurando eliminar sua dependência de processos *batch*. Para isso, entre outras adequações, é importante avaliar a possibilidade de substituição do algoritmo auto.arima, que é um método *batch*, para uma implementação de ARIMA que seja adaptada para uso com fluxos de dados, como a implementação de uma versão com suporte a janelas deslizantes (para eliminar a necessidade de conhecimento prévio de todos os dados da série), como sugerido nos estudos de Alberg e Last (2018) e de Saadallah et al. (2020) que fazem uso de ARIMA com suporte a janelas deslizantes e atualização automática e frequente de seus parâmetros.

Outra melhoria que deve ser avaliada para o método é em relação à sua interface com o *framework* MOA, a fim de eliminar a necessidade de utilização de arquivos externos nas interações entre a linguagem R e essa plataforma e, com isso, ampliar o desempenho geral do método.

Referências

ADHIKARI, Ratnadip; AGRAWAL, R. *An Introductory Study on Time series Modeling and Forecasting*. [S.l.: s.n.], 2013. ISBN 978-3-659-33508-2. Citado 7 vezes nas páginas 36, 38, 40, 41, 42, 46 e 47.

AHMED, Nesreen; ATIYA, Amir; GAYAR, Neamat; EL-SHISHINY, Hisham. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, v. 29, p. 594–621, 08 2010. Citado 4 vezes nas páginas 47, 194, 195 e 203.

ALBERG, Dima; LAST, Mark. Short-term load forecasting in smart meters with sliding window-based ARIMA algorithms. *Vietnam Journal of Computer Science*, v. 5, p. 241–249, 2018. Disponível em: <https://doi.org/10.1007/s40595-018-0119-7>. Citado 2 vezes nas páginas 77 e 170.

ALMEIDA, Ezilda; FERREIRA, Carlos; GAMA, João. Adaptive Model Rules from Data Streams. In: *ECML PKDD 2013: Machine Learning and Knowledge Discovery in Databases*. Springer, Berlin, Heidelberg, 2013. p. 480–492. Disponível em: <http://link.springer.com/10.1007/978-3-642-40988-2{_}31http://www.ecmlpkdd2013.org/wp-content/uploads/2013/07/251.> Citado 2 vezes nas páginas 72 e 188.

ALMEIDA, Ezilda; KOSINA, Petr; GAMA, João. Random rules from data streams. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2013. (SAC '13), p. 813–814. ISBN 9781450316569. Disponível em: <https://doi.org/10.1145/2480362.2480518>. Citado na página 69.

ARMSTRONG, J.S. *Long-range forecasting: from crystal ball to computer*. 2nd edition. ed. [S.l.]: Wiley, 1985. ISBN 9780471823605. Citado 2 vezes nas páginas 60 e 130.

ASSIMAKOPOULOS, V.; NIKOLOPOULOS, K. The theta model: a decomposition approach to forecasting. *International Journal of Forecasting*, v. 16, n. 4, p. 521 – 530, 2000. ISSN 0169-2070. The M3- Competition. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169207000000662>. Citado 4 vezes nas páginas 46, 56, 57 e 187.

BABCOCK, Brian; BABU, Shivnath; DATAR, Mayur; MOTWANI, Rajeev; WIDOM, Jennifer. "models and issues in data stream systems.". In: . [S.l.: s.n.], 2002. p. 1–16. Citado na página 166.

BARDDAL, J. P.; GOMES, H. M.; ENEMBRECK, F. A survey on feature drift adaptation. In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. [S.l.: s.n.], 2015. p. 1053–1060. Citado 2 vezes nas páginas 68 e 69.

BARTLETT, M. S. Some aspects of the time-correlation problem in regard to tests of significance. *Journal of the Royal Statistical Society*, [Wiley, Royal Statistical Society], v. 98, n. 3, p. 536–543, 1935. ISSN 09528385. Disponível em: <http://www.jstor.org/stable/2342284>. Citado na página 36.

BIFET, Albert; GAVALDA, Ricard; HOLMES, Geoff; PFAHRINGER, Bernhard. *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press, 2018. ISBN 978-0-262-03779-2. Disponível em: <https://moa.cms.waikato.ac.nz/book/>. Citado 8 vezes nas páginas 63, 66, 67, 71, 72, 73, 75 e 76.

BIFET, Albert; GAVALDà, Ricard. Learning from time-changing data with adaptive windowing. In: . [S.l.: s.n.], 2007. v. 7. Citado na página 65.

BIFET, Albert; HOLMES, Geoff; KIRKBY, Richard; PFAHRINGER, Bernhard. *MOA: Massive Online Analysis*. [S.l.], 2010. v. 11, 1601–1604 p. Citado 3 vezes nas páginas 72, 96 e 188.

BIFET, Albert; KIRKBY, Richard. *Data Stream Mining. A Practical Approach*. [S.l.], 2009. Disponível em: <https://www.cs.waikato.ac.nz/{~}abifet/MOA/StreamMining.p>. Citado na página 71.

BONTEMPI, Gianluca. *AMAR (All Methods Are Right)*. ULB, Université Libre de Bruxelles, Machine Learning Group, Belgium, 2018. Disponível em: <https://github.com/Mcompetitions/M4-methods/blob/master/043-gbonte/M4-AMAR-Bontempi-Method-Description.pdf>. Citado na página 50.

BOULEGANE, Dihia; BIFET, Albert; MADHUSUDAN, Giyyarpuram. Arbitrated dynamic ensemble with abstaining for time-series forecasting on data streams. In: . [S.l.: s.n.], 2019. Citado na página 33.

BOX, G.E.P.; JENKINS, G.M. *Time series analysis: forecasting and control*. [S.l.]: Holden-Day, 1976. (Holden-Day series in time series analysis and digital processing). ISBN 9780816211043. Citado na página 43.

BOX, G. E. P.; COX, D. R. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, [Royal Statistical Society, Wiley], v. 26, n. 2, p. 211–252, 1964. ISSN 00359246. Disponível em: <http://www.jstor.org/stable/2984418>. Citado na página 50.

BREIMAN, Leo. Random Forests. *Machine Learning*, v. 45, n. 1, p. 5–32, 2001. ISSN 1573-0565. Disponível em: <https://doi.org/10.1023/A:1010933404324>. Citado na página 69.

BRILLINGER, David. John w. tukey's work on time series and spectrum analysis. *The Annals of Statistics*, v. 30, 12 2002. Citado na página 36.

BROWN, R.G. *Statistical Forecasting for Inventory Control*. [S.l.]: McGraw-Hill, 1959. Citado na página 187.

BROWN, R.G. *Smoothing, forecasting and prediction of discrete time series*. [S.l.]: Englewood Cliffs, N.J., Prentice-Hall, Inc, 1963. Citado na página 46.

BROWNLEE, Jason. *Feature Selection for Time Series Forecasting with Python*. 2017. Disponível em: <https://machinelearningmastery.com/feature-selection-time-series-forecasting-python/>. Citado 2 vezes nas páginas 49 e 191.

CAI, Qingchao; XIE, Zhongle; ZHANG, Meihui; CHEN, Gang; JAGADISH, H. V.; OOI, Beng Chin. Effective temporal dependence discovery in time series data. *Proc. VLDB Endow.*, VLDB Endowment, v. 11, n. 8, p. 893–905, abr. 2018. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/3204028.3204033>. Citado na página 80.

CHAI, T.; DRAXLER, R. R. Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. *Geoscientific Model Development*, v. 7, n. 3, p. 1247–1250, 2014. Disponível em: <https://gmd.copernicus.org/articles/7/1247/2014/>. Citado na página 91.

CHEN, Tianqi; HE, Tong; BENESTY, Michael; KHOTILOVICH, Vadim; TANG, Yuan. *xgboost: Customized Extreme Gradient Boosting*. [S.l.], 2018. R package version 666.6.4.1. Disponível em: <https://CRAN.R-project.org/package=xgboost>. Citado 2 vezes nas páginas 59 e 187.

CLEMEN, Robert T. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, v. 5, n. 4, p. 559 – 583, 1989. ISSN 0169-2070. Disponível em: <http://www.sciencedirect.com/science/article/pii/0169207089900125>. Citado na página 46.

COOLEY, J. W.; TUKEY, J.W. An algorithm for the machine calculation of complex fourier series. *Math. Comp. 19 297-301. [Also in CWJWT II (1985) 651-658.]*, 1965. Citado na página 36.

DAIGO, Kato; TOMOHARU, Nagao. Stock prediction using multiple time series of stock prices and news articles. *2012 IEEE Symposium on Computers and Informatics, ISCI 2012*, 03 2012. Citado na página 54.

DATE, Sachin. *Understanding Partial Auto-Correlation*. Towards Data Science, 2019. Disponível em: <https://towardsdatascience.com/understanding-partial-auto-correlation-fa39271146ac>. Citado 2 vezes nas páginas 83 e 84.

DEAN, Jeffrey; CORRADO, Greg S.; MONGA, Rajat; CHEN, Kai; DEVIN, Matthieu; LE, Quoc V.; MAO, Mark Z.; RANZATO, Marc Aurelio; SENIOR, Andrew; TUCKER, Paul; YANG, Ke; NG, Andrew Y. Large Scale Distributed Deep Networks. In: *NIPS 2012: Neural Information Processing Systems*. [S.l.: s.n.], 2012. p. 1–11. Citado na página 79.

DEMŠAR, Janez. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, v. 7, n. Jan, p. 1–30, 2006. Citado 2 vezes nas páginas 146 e 148.

DICKEY, David A.; FULLER, Wayne A. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, [American Statistical Association, Taylor Francis, Ltd.], v. 74, n. 366, p. 427–431, 1979. ISSN 01621459. Citado na página 42.

DOMINGOS, Pedro; HULTEN, Geoff. Mining high-speed data streams. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2000. (KDD '00), p. 71–80. ISBN 1581132336. Disponível em: <https://doi.org/10.1145/347090.347107>. Citado 2 vezes nas páginas 69 e 72.

DONG, Ensheng; DU, Hongru; GARDNER, Lauren. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases*, Elsevier, v. 20, n. 5, p. 533–534, may 2020. ISSN 1473-3099. Disponível em: <https://doi.org/10.1016/S1473-3099(20)30120-1>. Citado na página 126.

DOORNIK, Jurgen; HENDRIK, David; CASTLE, Jennie. *Card (Calibrated average of rho and delta method)*. University of Oxford, UK, 2018. Disponível em: <https://github.com/Mcompetitions/M4-methods/blob/master/238-doornik/M4-Method-Description{_}Doornik.d>. Citado na página 50.

DUARTE, João; GAMA, João. Ensembles of adaptive model rules from high-speed data streams. In: FAN, Wei; BIFET, Albert; YANG, Qiang; YU, Philip S. (Ed.). *Proceedings of the 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. New York, New York, USA: PMLR, 2014. (Proceedings of Machine Learning Research, v. 36), p. 198–213. Disponível em: <http://proceedings.mlr.press/v36/duarte14.html>. Citado 3 vezes nas páginas 72, 74 e 188.

DUCHI, John; HAZAN, Elad; SINGER, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, JMLR.org, v. 12, n. null, p. 2121–2159, jul. 2011. ISSN 1532-4435. Citado 5 vezes nas páginas 43, 72, 75, 78 e 188.

DUONG, Quang-Huy; RAMAMPIARO, Heri; NØRVÅG, Kjetil. Applying temporal dependence to detect changes in streaming data. *Applied Intelligence*, v. 48, n. 12, p. 4805–4823, 2018. ISSN 1573-7497. Disponível em: <https://doi.org/10.1007/s10489-018-1254-7>. Citado na página 81.

ESLING, Philippe; AGON, Carlos. Time-series data mining. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 45, n. 1, dez. 2012. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/2379776.2379788>. Citado na página 36.

FERRER-TROYANO, Francisco J.; AGUILAR-RUIZ, Jesús S.; RIQUELME, José C. Incremental rule learning and border examples selection from numerical data streams. v. 11, n. 8, p. 1426–1439, aug 2005. <|>http://www.jucs.org/jucs$_1$1$_8$/$incremental_rule_learning_and$|.$Citadonapágina$69.

FILHO, Mario. *How To Predict Multiple Time Series With Scikit-Learn (With a Sales Forecasting Example)* . 2018. Disponível em: <http://mariofilho.com/how-to-predict-multiple-time-series-with-scikit-learn-with-sales-forecasting-example/>. Citado 2 vezes nas páginas 51 e 198.

FIORUCCI, José Augusto. *Time series forecasting: advances on Theta method*. Tese (Doutorado) — Universidade Federal de São Carlos - PPGEs/UFSCar, 2016. Disponível em: <https://repositorio.ufscar.br/bitstream/handle/ufscar/7399/TeseJAF.pdf?sequence=1>. Citado 2 vezes nas páginas 49 e 56.

FIORUCCI, Jose Augusto; LOUZADA, Francisco; YIQI, Bao. *forecTheta: Forecasting Time Series by Theta Models*. [S.l.], 2016. R package version 2.2. Disponível em: <https://CRAN.R-project.org/package=forecTheta>. Citado na página 62.

FIORUCCI, Jose A.; PELLEGRINI, Tiago R.; LOUZADA, Francisco; PETROPOULOS, Fotios; KOEHLER, Anne B. Models for optimising the theta method and their relationship to state space models. *International Journal of Forecasting*, v. 32, n. 4, p. 1151 – 1161, 2016. ISSN 0169-2070. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169207016300243>. Citado na página 56.

FLORES, João Henrique Ferreira; ENGEL, Paulo; PINTO, Rafael. Autocorrelation and partial autocorrelation functions to improve neural networks models on univariate time series forecasting. In: . [S.l.: s.n.], 2012. p. 1–8. ISBN 978-1-4673-1488-6. Citado na página 85.

FRIEDMAN, Milton. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, Taylor Francis, v. 32, n. 200, p. 675–701, 1937. Citado na página 149.

GAMA, João; MEDAS, Pedro; CASTILLO, Gladys; RODRIGUES, Pedro. Learning with drift detection. In: . [S.l.: s.n.], 2004. v. 8, p. 286–295. Citado na página 65.

GAMA, João; ŽLIOBAITė, Indrė; BIFET, Albert; PECHENIZKIY, Mykola; BOUCHACHIA, Abdelhamid. A survey on concept drift adaptation. *ACM Computing Surveys*, ACM, v. 46, n. 4, p. 1–37, mar 2014. ISSN 03600300. Disponível em: <http://dl.acm.org/citation.cfm?doid=2597757.2523813>. Citado 2 vezes nas páginas 63 e 188.

GAMA, João; KOSINA, Petr. Learning decision rules from data streams. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*. [S.l.]: AAAI Press, 2011. (IJCAI'11), p. 1255–1260. ISBN 9781577355144. Citado na página 69.

GARDNER, Everette S.; MCKENZIE, Ed. Forecasting trends in time series. *Manage. Sci.*, INFORMS, Linthicum, MD, USA, v. 31, n. 10, p. 1237–1246, out. 1985. ISSN 0025-1909. Disponível em: <https://doi.org/10.1287/mnsc.31.10.1237>. Citado na página 187.

GOMES, Heitor Murilo; BARDDAL, Jean Paul; BOIKO, Luis Eduardo; BIFET, Albert. Adaptive random forests for data stream regression. In: *ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Bruges (Belgium): [s.n.], 2018. ISBN 9782875870476. Disponível em: <https://www.researchgate.net/publication/329397374_Adaptive_random_forests_for_data_stream_regression>. Citado 3 vezes nas páginas 72, 73 e 188.

GOMES, Heitor Murilo; BIFET, Albert; READ, Jesse; BARDDAL, Jean Paul; ENEMBRECK, Fabrício; PFAHRINGER, Bernhard; HOLMES, Geoff; ABDESSALEM, Talel. Adaptive random forests for evolving data stream classification. *Machine Learning*, p. 1–27, 06 2017. Citado na página 73.

GOMES, Heitor Murilo; READ, Jesse; BIFET, Albert; BARDDAL, Jean Paul; GAMA, João. Machine learning for streaming data: State of the art, challenges, and opportunities. *SIGKDD Explor. Newsl.*, Association for Computing Machinery, New York, NY, USA, v. 21, n. 2, p. 6–22, nov. 2019. ISSN 1931-0145. Disponível em: <https://doi.org/10.1145/3373464.3373470>. Citado 2 vezes nas páginas 75 e 76.

GOODRICH, R. The forecast pro methodology. *International Journal of Forecasting*, v. 16, p. 451–476, 2000. Citado na página 47.

GOOIJER, Jan G. De; HYNDMAN, Rob. 25 years of time series forecasting. *International Journal of Forecasting*, v. 22, p. 443–473, 02 2006. Citado 3 vezes nas páginas 43, 45 e 46.

GOUK, Henry; PFAHRINGER, Bernhard; FRANK, Eibe. *Stochastic Gradient Trees*. 2019. Citado na página 74.

GUNAWARDANA, Asela; MEEK, Christopher; XU, Puyang. A model for temporal dependencies in event streams. *Advances in Neural Information Processing Systems*, 01 2011. Citado na página 81.

HAMNER, Ben; FRASCO, Michael. *Metrics: Evaluation Metrics for Machine Learning*. [S.l.], 2018. R package version 0.1.4. Disponível em: <https://CRAN.R-project.org/package=Metrics>. Citado 2 vezes nas páginas 131 e 186.

HANNAN, E.J. Introductory theory. In: _____. *Multiple Time Series*. John Wiley Sons, Ltd, 1970. cap. 1, p. 2–31. ISBN 9780470316429. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470316429.ch1>. Citado na página 31.

HOLT, Charles C. Forecasting seasonals and trends by exponentially weighted moving averages. *Office of Naval Reseach (ONR) Memorandum No. 52, Carnegie Institute of Technology, Graduate school of Industrial Administration, Pittsburgh*, 1957. Citado 2 vezes nas páginas 187 e 192.

HYNDMAN, Rob; ATHANASOPOULOS, George; BERGMEIR, Christoph; CACERES, Gabriel; CHHAY, Leanne; O'HARA-WILD, Mitchell; PETROPOULOS, Fotios; RAZBASH, Slava; WANG, Earo; YASMEEN, Farah. *ets - Exponential Smoothing State Space Model*. 2019. Disponível em: <https://www.rdocumentation.org/packages/forecast/versions/8.5/topics/ets>. Citado 3 vezes nas páginas 61, 62 e 187.

HYNDMAN, Rob; ATHANASOPOULOS, George; BERGMEIR, Christoph; CACERES, Gabriel; CHHAY, Leanne; O'HARA-WILD, Mitchell; PETROPOULOS, Fotios; RAZBASH, Slava; WANG, Earo; YASMEEN, Farah. *forecast: Forecasting functions for time series and linear models*. [S.l.], 2019. R package version 8.9.0.9000. Disponível em: <http://pkg.robjhyndman.com/forecast>. Citado 8 vezes nas páginas 15, 61, 62, 82, 97, 186, 188 e 192.

HYNDMAN, Rob; ATHANASOPOULOS, George; BERGMEIR, Christoph; CACERES, Gabriel; CHHAY, Leanne; O'HARA-WILD, Mitchell; PETROPOULOS, Fotios; RAZBASH, Slava; WANG, Earo; YASMEEN, Farah. *forecast.stl - Forecasting Using Stl Objects*. 2019. Disponível em: <https://www.rdocumentation.org/packages/forecast/versions/8.5/topics/forecast.stl>. Citado na página 187.

HYNDMAN, Rob; ATHANASOPOULOS, George; BERGMEIR, Christoph; CACERES, Gabriel; CHHAY, Leanne; O'HARA-WILD, Mitchell; PETROPOULOS, Fotios; RAZBASH, Slava; WANG, Earo; YASMEEN, Farah. *rwf - Naive and Random Walk Forecasts*. 2019. Disponível em: <http://pkg.robjhyndman.com/forecast/reference/naive.html>. Citado na página 187.

HYNDMAN, Rob John; BILLAH, Md Baki. Unmasking the theta method. *International Journal of Forecasting*, Elsevier, v. 19, n. 2, p. 287 – 290, 2003. ISSN 0169-2070. Citado 3 vezes nas páginas 56, 57 e 187.

HYNDMAN, Rob; WANG, Earo; KANG, Yanfei; TALAGALA, Thiyanga. *tsfeatures: Time Series Feature Extraction*. [S.l.], 2019. R package version 0.1. Disponível em: <https://github.com/robjhyndman/tsfeatures/>. Citado 2 vezes nas páginas 158 e 203.

HYNDMAN, Rob; YANG, Yangzhuoran. *tsdl: Time Series Data Library*. [S.l.], 2020. Https://finyang.github.io/tsdl/, https://github.com/FinYang/tsdl. Citado 2 vezes nas páginas 127 e 128.

HYNDMAN, Rob J. *Forecasting: Principles and Practice*. 2014. Disponível em: <https://robjhyndman.com/uwafiles/fpp-notes.pdf>. Citado na página 192.

HYNDMAN, Rob J.; ATHANASOPOULOS, George. *Forecasting: Principles and Practice, 2nd edition*. 2018. Disponível em: <https://otexts.com/fpp2/>. Citado 12 vezes nas páginas 15, 37, 38, 40, 42, 43, 44, 46, 81, 82, 83 e 121.

HYNDMAN, Rob J; KHANDAKAR, Yeasmin. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, v. 26, n. 3, p. 1–22, 2008. Disponível em: <http://www.jstatsoft.org/article/view/v027i03>. Citado 3 vezes nas páginas 43, 188 e 192.

HYNDMAN, Rob J.; KOEHLER, Anne B. Another look at measures of forecast accuracy. *International Journal of Forecasting*, v. 22, n. 4, p. 679 – 688, 2006. ISSN 0169-2070. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169207006000239>. Citado 4 vezes nas páginas 87, 88, 89 e 90.

IKONOMOVSKA, Elena; GAMA, João; DžEROSKI, Sašo. Online tree-based ensembles and option trees for regression on evolving data streams. *Neurocomputing*, v. 150, 11 2014. Citado 2 vezes nas páginas 73 e 74.

IKONOMOVSKA, Elena; GAMA, João; ŽENKO, Bernard; DžEROSKI, Sašo. Speeding-up hoeffding-based regression trees with options. In: . [S.l.: s.n.], 2011. p. 537–544. Citado 2 vezes nas páginas 72 e 188.

IKONOMOVSKA, Elena; GAMA, João; DzEROSKI, Sašo. Learning model trees from evolving data streams. *Data Min. Knowl. Discov.*, Kluwer Academic Publishers, USA, v. 23, n. 1, p. 128–168, jul. 2011. ISSN 1384-5810. Disponível em: <https://doi.org/10.1007/s10618-010-0201-y>. Citado 2 vezes nas páginas 72 e 188.

JACOBY, William G. Loess:: a nonparametric, graphical tool for depicting relationships between variables. *Electoral Studies*, v. 19, n. 4, p. 577 – 613, 2000. ISSN 0261-3794. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0261379499000281>. Citado na página 39.

JAIN, Chaman L. *Fundamentals of Demand Planning and Forecasting*. 3rd edition. ed. [S.l.]: Graceway Publishing Company, Inc., 2017. ISBN 978-0-9839413-2-3. Citado 4 vezes nas páginas 40, 41, 42 e 45.

KARIM, Raimi. *10 Gradient Descent Optimisation Algorithms + Cheat Sheet*. 2018. Disponível em: <https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9>. Citado 2 vezes nas páginas 78 e 79.

KASSAMBARA, Alboukadel; MUNDT, Fabian. *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*. [S.l.], 2017. R package version 1.0.5. Disponível em: <https://CRAN.R-project.org/package=factoextra>. Citado na página 208.

KHAN, Arijit; YAN, Xifeng; TAO, Shu; ANEROUSIS, Nikos. Workload characterization and prediction in the cloud: A multiple time series approach. In: *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012*. [S.l.: s.n.], 2012. p. 1287–1294. ISBN 9781467302685. Citado na página 54.

KIFER, Daniel; BEN-DAVID, Shai; GEHRKE, Johannes. Detecting change in data streams. In: *Proceedings of the Thirtieth International Conference on Very Large Data Bases*. [S.l.: s.n.], 2004. v. 30, p. 180–191. Citado na página 66.

KOLMOGOROFF, A. Giorm Ist Ipal Attuari. v. 4, n. 83, 1933. Citado na página 36.

KREMPL, Georg; ZLIOBAITE, Indre; BRZEZIńSKI, Dariusz; HüLLERMEIER, Eyke; LAST, Mark; LEMAIRE, Vincent; NOACK, Tino; SHAKER, Ammar; SIEVI, Sonja; SPILIOPOULOU, Myra; STEFANOWSKI, Jerzy. Open challenges for data stream mining research. *SIGKDD Explor. Newsl.*, Association for Computing Machinery, New York, NY, USA, v. 16, n. 1, p. 1–10, set. 2014. ISSN 1931-0145. Disponível em: <https://doi.org/10.1145/2674026.2674028>. Citado na página 77.

KWIATKOWSKI, Denis; PHILLIPS, Peter C.B.; SCHMIDT, Peter; SHIN, Yongcheol. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, v. 54, n. 1, p. 159 – 178, 1992. ISSN 0304-4076. Disponível em: <http://www.sciencedirect.com/science/article/pii/030440769290104Y>. Citado na página 44.

LAST, Mark. Online classification of nonstationary data streams. *Intell. Data Anal.*, v. 6, p. 129–147, 07 2002. Citado na página 77.

LÊ, Sébastien; JOSSE, Julie; HUSSON, François. FactoMineR: A package for multivariate analysis. *Journal of Statistical Software*, v. 25, n. 1, p. 1–18, 2008. Citado na página 208.

LEI, Zhang. Nonlinear prediction of exchange rate: A new approach to multiple time series analysis. In: *International Conference on Management Science and Engineering - Annual Conference Proceedings*. [S.l.: s.n.], 2013. p. 1798–1803. ISBN 9781479904716. ISSN 21551847. Citado na página 55.

LU, J.; LIU, A.; DONG, F.; GU, F.; GAMA, J.; ZHANG, G. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, v. 31, n. 12, p. 2346–2363, 2019. Citado 8 vezes nas páginas 15, 63, 64, 65, 66, 67, 68 e 70.

MAKRIDAKIS, Spyros. *A Survey of Time Series*. [S.l.], 1976. v. 44, n. 1, 29–70 p. Disponível em: <http://hdl.handle.net/11728/6326>. Citado 5 vezes nas páginas 36, 38, 40, 42 e 46.

MAKRIDAKIS, Spyros; HIBON, Michele. The m3-competition: Results, conclusions and implications. *International Journal of Forecasting*, v. 16, p. 451–476, 10 2000. Citado 6 vezes nas páginas 32, 46, 55, 123, 130 e 185.

MAKRIDAKIS, Spyros; SPILIOTIS, Evangelos; ASSIMAKOPOULOS, Vassilios. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, v. 34, n. 4, p. 802 – 808, 2018. ISSN 0169-2070. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169207018300785>. Citado 13 vezes nas páginas 15, 21, 32, 38, 39, 47, 55, 82, 124, 185, 195, 196 e 197.

MAKRIDAKIS, Spyros; SPILIOTIS, Evangelos; ASSIMAKOPOULOS, Vassilios. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, Public Library of Science, v. 13, n. 3, p. 1–26, 03 2018. Disponível em: <https://doi.org/10.1371/journal.pone.0194889>. Citado 3 vezes nas páginas 131, 194 e 195.

MAKRIDAKIS, Spyros; SPILIOTIS, Evangelos; ASSIMAKOPOULOS, Vassilis. *The M5 Accuracy competition: Results, findings and conclusions*. 2020. Disponível em: <https://www.researchgate.net/publication/344487258_The_M5_Accuracy_competition_Results_findings_and_conclusions>. Citado 2 vezes nas páginas 32 e 125.

MAKRIDAKIS, Spyros G.; WHEELWRIGHT, Steven C.; HYNDMAN, Rob J. *Forecasting: Methods and Applications*. 3rd edition. ed. New York: Wiley, 1998. Citado na página 61.

MCKENZIE, Eddie; GARDNER, Everette. Damped trend exponential smoothing: A modelling viewpoint. *International Journal of Forecasting*, v. 26, p. 661–665, 10 2010. Citado na página 47.

MONTERO-MANSO, Pablo; ATHANASOPOULOS, George; HYNDMAN, Rob J; TALAGALA, Thiyanga S. *FFORMA: Feature-based forecast model averaging*. [S.l.], 2018. Citado 3 vezes nas páginas 53, 59 e 211.

MONTERO-MANSO, Pablo; ATHANASOPOULOS, George; HYNDMAN, Rob J; TALAGALA, Thiyanga S. *M4metalearning*. University of A Coruña (Spain) e Monash University (Austrália e Sri Lanka), 2018. Disponível em: <https://github.com/Mcompetitions/M4-methods/blob/master/245-pmontman/M4-Method-Description.pdf>. Citado 3 vezes nas páginas 59, 187 e 213.

MORALES, Gianmarco De Francisci; BIFET, Albert; KHAN, Latifur; GAMA, Joao; FAN, Wei. Iot big data stream mining. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 2119–2120. ISBN 9781450342322. Disponível em: <https://doi.org/10.1145/2939672.2945385>. Citado na página 31.

NEMENYI, Peter. Distribution-free multiple comparisons. In: INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210. *Biometrics*. [S.l.], 1962. v. 18, n. 2, p. 263. Citado na página 149.

NIELSEN, Ulrik D.; BRODTKORB, Astrid H.; JENSEN, Jørgen J. Response predictions using the observed autocorrelation function. *Marine Structures*, v. 58, p. 31 – 52, 2018. ISSN 0951-8339. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0951833917301284>. Citado na página 80.

NIKOLOPOULOS, Konstantinos; ASSIMAKOPOULOS, Vassilis; BOUGIOUKOS, Nikolaos; LITSA, Akrivi; PETROPOULOS, Fotios. The theta model: An essential

forecasting tool for supply chain planning. *Lecture Notes in Electrical Engineering*, v. 123, p. 431–437, 01 2011. Citado na página 56.

OSTERTAGOVA, Eva; OSTERTAG, Oskar. Forecasting using simple exponential smoothing method. *Acta Electrotechnica et Informatica*, v. 12, p. 62–66, 12 2012. Citado na página 192.

PAGE, E. S. Continuous inspection schemes. *Biometrika*, [Oxford University Press, Biometrika Trust], v. 41, n. 1/2, p. 100–115, 1954. ISSN 00063444. Disponível em: <http://www.jstor.org/stable/2333009>. Citado na página 66.

PARZEN, Emanuel. Ararma models for time series analysis and forecasting. *Journal of Forecasting*, v. 1, p. 67 – 82, 01 1982. Citado na página 47.

PAWLIKOWSKI, Maciej; CHOROWSKA, Agata. Weighted ensemble of statistical models. *International Journal of Forecasting*, v. 36, n. 1, p. 93 – 97, 2020. ISSN 0169-2070. M4 Competition. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169207019301190>. Citado 3 vezes nas páginas 50, 60 e 61.

PAWLIKOWSKI, Maciej; CHOROWSKA, Agata; YANCHUK, Olena. *Weighted Ensemble of Statistical Models*. ProLogistica Soft, Poland, 2018. Disponível em: <https://github.com/Mcompetitions/M4-methods/blob/master/237-prologistica/M4-Method-Description-ProLogistica_Soft.pdf>. Citado 2 vezes nas páginas 51 e 60.

PRABHAKARAN, Selva. *ARIMA Model - Complete Guide to Time Series Forecasting in Python | ML+*. 2019. Disponível em: <https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/>. Citado na página 42.

QIU, Debin. *aTSA: Alternative Time Series Analysis*. [S.l.], 2015. R package version 3.1.2. Disponível em: <https://CRAN.R-project.org/package=aTSA>. Citado na página 187.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2018. Disponível em: <https://www.R-project.org/>. Citado 2 vezes nas páginas 15 e 39.

RAMíREZ-GALLEGO, Sergio; KRAWCZYK, Bartosz; GARCíA, Salvador; WOźNIAK, Michał; HERRERA, Francisco. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, v. 239, p. 39 – 57, 2017. ISSN 0925-2312. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231217302631>. Citado 2 vezes nas páginas 15 e 65.

RODRIGUES, Pedro; GAMA, João. A system for analysis and prediction of electricity-load streams. *Intell. Data Anal.*, v. 13, p. 477–496, 01 2009. Citado na página 80.

RUDER, Sebastian. *An overview of gradient descent optimization algorithms*. [S.l.], 2017. Disponível em: <http://caffe.berkeleyvision.org/tutorial/solver.html>. Citado 2 vezes nas páginas 78 e 79.

SAADALLAH, A.; MOREIRA-MATIAS, L.; SOUSA, R.; KHIARI, J.; JENELIUS, E.; GAMA, J. Bright—drift-aware demand predictions for taxi networks. *IEEE Transactions on Knowledge and Data Engineering*, v. 32, n. 2, p. 234–245, 2020. Citado na página 170.

SAIGAL, S.; MEHROTRA, D. Performance comparison of time series data using predictive data mining techniques. *Advances in Information Mining*, v. 4, Issue 1, p. 57–66, 2012. ISSN 0975-3265 & E-ISSN 0975-9093. Citado na página 87.

SLUTZKY, Eugen. The summation of random causes as the source of cyclic processes. *Econometrica*, [Wiley, Econometric Society], v. 5, n. 2, p. 105–146, 1937. ISSN 00129682, 14680262. Citado na página 42.

SMYL, Slawek. *Exponential Smoothing - Recurrent Neural Network (ES-RNN) Hybrid Models*. Uber Technologies, USA., 2018. Disponível em: <https://github.com/Mcompetitions/M4-methods/blob/master/118-slaweks17/ES_RNN_SlawekSmyl.pdf>. Citado 2 vezes nas páginas 47 e 58.

SMYL, Slawek; RANGANATHAN, Jai; PASQUA, Andrea. *M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model*. Uber Technologies, USA., 2018. Disponível em: <https://eng.uber.com/m4-forecasting-competition/>. Citado 4 vezes nas páginas 51, 54, 58 e 192.

SOBHY, Dalia. *Continuous evaluation framework for software architectures: an IoT case*. Tese (Tese de Doutorado) — University of Birmingham, 2019. Disponível em: <https://etheses.bham.ac.uk/id/eprint/9312/7/Sobhy2019PhD.pdf>. Citado 3 vezes nas páginas 72, 73 e 188.

STOJANOVA, Daniela. *Considering Autocorrelation in Predictive Models*. Tese (Doutorado) — Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, December 2012. Doctoral Dissertation. Citado na página 80.

TAIEB, Souhaib Ben; SORJAMAA, Antti; BONTEMPI, Gianluca. Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing*, v. 73, n. 10, p. 1950 – 1957, 2010. ISSN 0925-2312. Subspace Learning / Selected papers from the European Symposium on Time Series Prediction. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231210001013>. Citado na página 54.

TALAGALA, Thiyanga S; HYNDMAN, Rob J; ATHANASOPOULOS, George. *Meta-learning how to forecast time series*. [S.l.], 2018. Disponível em: <http://business.monash.edu/econometrics-and-business-statistics/research/publications>. Citado na página 192.

TASHMAN, Leonard J. Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting*, v. 16, n. 4, p. 437 – 450, 2000. ISSN 0169-2070. The M3- Competition. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169207000000650>. Citado na página 61.

TAYLOR, Sean J; LETHAM, Benjamin. Forecasting at scale. PeerJ Inc., sep 2017. ISSN 2167-9843. Disponível em: <https://peerj.com/preprints/3190/>. Citado na página 186.

TROTTA, Belinda. *Convolutional Neural Networks for Time Series (CNN-TS)*. Australia, 2018. Disponível em: <https://github.com/Mcompetitions/M4-methods/blob/master/211-btrotta/M4-Method-Description-CNN-TS.pdf>. Citado 3 vezes nas páginas 49, 50 e 52.

WALGAMPAYA, C.; KANTARDZIC, M. Selection of distributed sensors for multiple time series prediction. In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. [S.l.: s.n.], 2006. p. 3152–3158. ISSN 2161-4407. Citado na página 55.

WALKER, Gilbert. On periodicity in series of related terms. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, The Royal Society, v. 131, n. 818, p. 518–532, 1931. ISSN 09501207. Citado na página 42.

WALTERS, Troy. *Time Series Analysis in R Part 2: Time Series Transformations*. 2018. Disponível em: <https://datascienceplus.com/time-series-analysis-in-r-part-2-time-series-transformations/>. Citado na página 191.

WERNER, Liane; RIBEIRO, JosÃ©Luis Duarte. PrevisÃde demanda: uma aplicaÃ§Ãdos modelos Box-Jenkins na Ã¡rea de assistÃtÃde computadores pessoais. *GestÃ ProduÃ§Ã*, scielo, v. 10, p. 47 – 67, 04 2003. ISSN 0104-530X. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2003000100005&nrm=iso>. Citado na página 81.

WIJAYA, Galih Praja; HANDIAN, Dendi; NASRULLOH, Imam Fachmi; RIZA, Lala Septem; MEGASARI, Rani; JUNAETI, Enjun. *gradDescent: Gradient Descent for Regression Tasks*. [S.l.], 2018. R package version 3.0. Disponível em: <https://CRAN.R-project.org/package=gradDescent>. Citado na página 217.

WINTERS, Peter R. Forecasting sales by exponentially weighted moving averages. *Management Science*, v. 6, n. 3, p. 324–342, 1960. Disponível em: <https://doi.org/10.1287/mnsc.6.3.324>. Citado na página 187.

YULE, George Udny. On a method of investigating periodicities disturbed series, with special reference to Wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, v. 226, p. 267–298, 1927. Citado na página 42.

ZLIOBAITE, Indre. Learning under concept drift: an overview. *CoRR*, abs/1010.4784, 2010. Disponível em: <http://arxiv.org/abs/1010.4784>. Citado 2 vezes nas páginas 15 e 64.

ŽLIOBAITė, Indrė; BIFET, Albert; READ, Jesse; PFAHRINGER, Bernhard; HOLMES, Geoff. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, v. 98, n. 3, p. 455–482, 2015. ISSN 1573-0565. Disponível em: <https://doi.org/10.1007/s10994-014-5441-4>. Citado 4 vezes nas páginas 79, 80, 100 e 216.

Apêndices

APÊNDICE A – Estudos e Experimentos para a Estruturação Geral de Método de Predição

Descrição Geral

Os estudos e experimentos descritos neste apêndice tiveram como objetivo estabelecer a estrutura geral para um **método de predição de séries temporais univariadas** utilizando algoritmos de mineração de fluxos de dados capaz de apresentar um desempenho comparável ou superior ao obtido por algoritmos do estado-da-arte, em especial pelos métodos estatísticos ou outros métodos que tenham obtido destaque em competições internacionais de predições de séries dessa natureza.

Os estudos incluem, de forma geral, a busca de informação para a formação de uma compreensão básica sobre os principais conceitos envolvidos no processo de predição de séries temporais univariadas. A identificação de principais técnicas estatísticas associadas ao processo, bem como o estudo de algoritmos de mineração de fluxos de dados e a forma como utilizá-los em processos de regressão fazem parte dessa análise.

Para os experimentos foram selecionadas bases de dados de séries temporais das competições M3 (MAKRIDAKIS; HIBON, 2000) e M4 (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a), conhecidas por fazerem parte das *M-Competitions*, ou *Makridakis-Competitions* (https://mofc.unic.ac.cy/). Elas se referem a séries amplamente divulgadas e discutidas no meio acadêmico, sendo possível encontrar vários artigos na literatura que fazem referência a tais competições. No caso da competição M4, ela é também uma competição amplamente divulgada pelo *International Institute of Forecasting* (https://forecasters.org/) em sua publicação oficial denominada *International Journal of Forecasting* (https://ijf.forecasters.org/).

Então, com o objetivo de estabelecer tal método de predição, foram realizados experimentos em duas fases, conforme descrito a seguir.

Na **FASE 1** dos experimentos, foram realizadas análises com 1.428 séries temporais mensais da competição M3, fazendo testes com subconjuntos de 1.045 (séries com mais de 81 registros para treinamento), 1086 (séries mensais de M3 com mais de 81 registros no total, incluindo dados da série e 18 registros de testes) e o conjunto completo com as 1428 séries.

Na **FASE 2**, o banco de dados da competição M4 foi utilizado por possuir um maior número de séries com o propósito de validar e aprimorar o método de predição estabelecido durante os experimentos da Fase 1 do estudo. Nessa fase, foram realizadas análises com diferentes subconjuntos das 48.000 séries mensais de M4, como experimentos com 100, 500, 999, 1.000, 5.000, 48.000 séries.

Ao final dos experimentos das Fases 1 e 2, os resultados de predição obtidos por uma combinação de métodos selecionados ainda não foram capazes de superar de forma expressiva a acurácia obtida pelos métodos participantes da competição M4 melhores colocados no processamento de séries mensais. Mas, os resultados conseguiram se mostrar melhores que os apresentados pelo competidor melhor colocado na categoria *machine learning*. Além disso, permitiram identificar que existe um potencial para superar os resultados obtidos até o momento se ajustes no processo de seleção de algoritmos forem realizados de forma a atribuir, a AdaGrad (algoritmo de mineração de fluxo de dados que apresentou melhores resultados), as séries em que seja capaz de apresentar menores erros de predição. Com isso, sugere-se que o método proposto é competitivo e merece estudos mais detalhados.

Uma parte comum dos experimentos foi o processamento dos *datasets* das séries em *softwares* de Aprendizagem de Máquina (*Machine Learning*) capazes de realizar o processo de treinamento/criação de modelo de aprendizagem (*fitting*) e predição (*forecasting*). Para isso, os principais softwares utilizados nos experimentos incluíram:

- Software R, executado a partir da IDE RStudio:
  - Utilizado principalmente nos processos de:
    - *Feature engineering* (criação de características/atributos).
    - Geração de arquivos para processamento.
    - Predições com algoritmos convencionais (não orientados a fluxos de dados).
    - Cálculo de erros.
    - Simulação de *ensemble* (combinação de algoritmos).
  - Principais pacotes utilizados para funções de predição e cálculos/métricas de erros:
    - *forecast* (HYNDMAN et al., 2019b).
    - *Metrics* (HAMNER; FRASCO, 2018).
  - Principal algoritmo estatístico utilizado para a predição:
    - *forecast::auto.arima*.
  - Outros métodos utilizados para testes de predição:
    - Prophet (Facebook) (TAYLOR; LETHAM, 2017).

    * Método M4metalearning (MONTERO-MANSO et al., 2018b).
        · O método inclui os algoritmos auto.arima, ETS, NNETAR, TBATS, STLM-AR, RW-DRIFT, THETA, NAIVE, SEASONAL NAIVE.
        · Utiliza os pacotes para o R: *"pmontan/tsfeatures"* e *"pmontan/xgboost"*, que são versões customizadas dos pacotes *tsfeatures* (MONTERO-MANSO et al., 2018b) e *xgboost* (CHEN et al., 2018).
        · O método foi alterado para a inclusão de AdaGrad em sua lista de algoritmos.

– Algoritmos usados no processo de criação de características para alguns experimentos (com destaque para ARIMA, que foi selecionado como algoritmo para o processo de *feature engineering* para o método final):

    * **SES** (*Simple Exponential Smoothing*)
        · Referências: (BROWN, 1959), (HOLT, 1957), (WINTERS, 1960)
        · Função *forecast::ses* (pacote *forecast*)
    * **Holt** (*Holt's Two-parameter Exponential Smoothing*)
        · Referências: (HOLT, 1957)
        · Função *aTSA::Holt* (pacote *aTSA*, (QIU, 2015))
    * **Winters** (*Winters Three-parameter Smoothing*)
        · Referências: (WINTERS, 1960)
        · Função *aTSA::Winters* (pacote *aTSA*)
    * **Damped** (*Holt's linear method with damped trend*)
        · Referências: (HOLT, 1957), (GARDNER; MCKENZIE, 1985)
        · Função *forecast::holt()*, com parâmetro *damped=true* (pacote *forecast*)
    * **Random Walk forecast**
        · Função *forecast::rwf* , (HYNDMAN et al., 2019d), (pacote *forecast*)
    * **Seasonal Naive**
        · Função *forecast::snaive*, (HYNDMAN et al., 2019d), (pacote *forecast*)
    * **Theta method**
        · Referências: (ASSIMAKOPOULOS; NIKOLOPOULOS, 2000), (HYNDMAN; BILLAH, 2003)
        · Função *forecast::thetaf* (pacote *forecast*)
    * **ETS** (*Exponential smoothing state space model*)
        · Função *forecast::ets*, (HYNDMAN et al., 2019a), (pacote *forecast*)
    * **STLM** (STL *decomposition*)
        · Função *forecast::stlm*, (HYNDMAN et al., 2019c), (pacote *forecast*)
    * **ARIMA**

        - Referências: (HYNDMAN et al., 2019b), (HYNDMAN; KHANDAKAR, 2008)
        - Função *forecast::auto.arima* (pacote *forecast*)

- *Software* MOA (*Massive Online Analysis*) (BIFET et al., 2010) - *software* para mineração de fluxos de dados (*data stream mining*).
  - Algoritmos de aprendizagem (*learners*) de regressão utilizados para testes de predição:
    - **AdaGrad** (*moa.classifiers.functions.AdaGrad*, (BIFET et al., 2010))
      - *Adaptive Gradient Algorithm* (Algoritmo de Gradiente Adaptativo)
      - Referência: (DUCHI; HAZAN; SINGER, 2011)
    - **AMRules Regressor** (*moa.classifiers.rules.AMRulesRegressor,* (BIFET et al., 2010))
      - *Adaptive Model Rules regressor*
      - Referência: (ALMEIDA; FERREIRA; GAMA, 2013)
    - **ARF-Reg** (*moa.classifiers.meta.AdaptiveRandomForestRegressor*, (BIFET et al., 2010))
      - *Adaptive Random Forest for Data Stream Regression*
      - Referência: (GOMES et al., 2018)
    - **FIMT-DD** (*moa.classifiers.trees.FIMTDD*, (BIFET et al., 2010))
      - *Fast Incremental Model Tree with Drift Detection*
      - Referência: (IKONOMOVSKA; GAMA; DzEROSKI, 2011)
    - **ORTO** (*moa.classifiers.trees.ORTO*, (BIFET et al., 2010))
      - *Online Option Trees for Regression*
      - Referência: (IKONOMOVSKA et al., 2011)
    - **Random AMRules** (*moa.classifiers.rules.meta.RandomAMRules*, (BIFET et al., 2010))
      - Referência: (DUARTE; GAMA, 2014)
    - **RandomRules** (*moa.classifiers.meta.RandomRules*, (BIFET et al., 2010))
      - class *moa.classifiers.meta.RandomRules in MOA*
    - **FadingTargetMean** (*moa.classifiers.rules.functions.FadingTargetMean,* (BIFET et al., 2010))
      - Referência: (SOBHY, 2019) explica que esse método de predição, baseado na média de *targets* de instâncias treinadas, usa um fator de esquecimento (*fading*) de forma a dar mais importância a dados mais recentes e menor importância a valores mais distantes das séries. Cita, ainda, as referências (BIFET et al., 2010) e (GAMA et al., 2014).

Entre as principais atividades realizadas durante os experimentos, pode-se citar:

- Processo de **Criação/Seleção de Características** (*feature engineering*).
  - Nesse processo, foram utilizados metadados baseados na execução de algoritmos de regressão, em especial o ARIMA, pelo uso do método *forecast::auto.arima* no R.
  - Ao usar ARIMA para esse processo, foi utilizado o conceito de ***metalearning*** uma vez que, dessa forma, identificam-se informações ou tendências não perceptíveis diretamente nos valores das observações da série.
- Predições de séries temporais utilizando métodos estatísticos e algoritmos de mineração de fluxos de dados.
- Análise de erros de predição com base em diferentes métricas, focando especialmente no uso de sMAPE (*Symmetric Mean Absolute Percentage Error*).
- Experimentos com uso dos conceitos de combinação e fusão a fim de verificar se a utilização conjunta de diferentes técnicas de processamento com AdaGrad poderia trazer melhores resultados para o processo de predição.
- Estudos de processos de seleção baseadas no uso de diferentes critérios de seleção de algoritmos, em especial tomando como base o valor de erro de predição gerado pelos algoritmos em etapa de treinamento e/ou validação.
- Experimentos com diferentes métodos para a extração de características de séries temporais e estudos de conceitos de meta-aprendizagem.

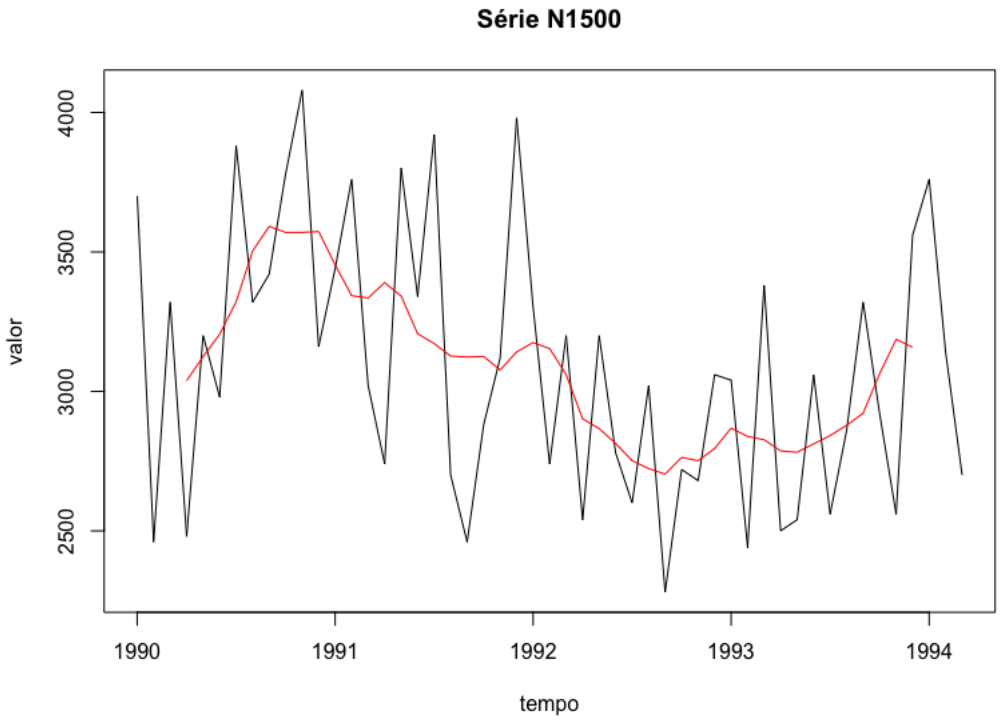
As seções a seguir descrevem, de forma geral, os experimentos realizados em cada fase. Ao final da descrição geral de cada experimento é disponibilizado um *link* para um documento externo que o descreve em mais detalhes e com as informações organizadas conforme os itens a seguir:

- Descrição do Experimento.
- Séries temporais / Multisséries.
- *Feature engineering* (engenharia de características).
- Pré-processamento.
- Processamento.
- Análise dos resultados obtidos.
- Conclusões do experimento.

A.1 Descrição dos Experimentos - FASE 1

Nessa fase, os experimentos foram realizados com um conjunto de 1.428 séries temporais mensais extraídas dos dados da competição M3. A Figura 37 apresenta um exemplo de uma série temporal de M3.

Figura 37 – Exemplo de uma série temporal de M3, com observações da série em preto e curva suavizada por média móvel (6 meses) em vermelho, sem normalização.



Fonte: Autoria própria

As séries mensais desse conjunto M3 representam informações extraídas de diferentes domínios (Microeconomia, Indústria, Macroeconomia, Finanças, Dados Demográficos e Outros) e possuem as seguintes características estatísticas:

- média = 99,34 eventos por série.
- mínimo = 48 eventos por série.
- máximo = 126 eventos por série.
- desvio padrão = 28,50.

Além das informações básicas da série (geralmente Mês/Ano e Observações - eventos da série em determinado mês) algoritmos para criação de *features* geralmente são utilizados na análise de séries temporais e incluem os cálculos de:

- Atrasos (*Lags*) - deslocamento das observações da série para diferentes intervalos.
- Differenças (*Difference, ou Diffs*) - diferenças entre as observações de um mês e outro.
- Médias móveis (*Moving averages*) - médias das observações da série em diferentes meses.
- Variação percentual (*Percent change*) - variação percentual do número de observações de um mês em relação a outro.
- Log (*log difference*) - valores transformados com o uso de função logarítmica.
- Entre outros.

Para o uso de algoritmos de aprendizagem na análise de séries temporais univariadas, (BROWNLEE, 2017) descreve que é preciso trabalhar na criação de características (*feature engineering*) uma vez que uma série temporal univariada se apresenta como uma simples sequência de eventos, sem muitas informações adicionais. Para a análise das séries, sugere que a série seja transformada em estacionária (realizar, por exemplo, um *diff(12)* da série, visto que os dados são mensais e a sazonalidade é anual). Depois recomenda efetuar a criação de um correlograma dos dados da série (*autocorrelation plot*), para verificar a relevância de cada *feature* (no caso, *lagged features*) em relação à classe (*target*) da série. Nesse caso, pode-se dizer que a correlação identificada é entre uma variável (no caso os dados da série em determinado momento) e o seu valor em períodos anteriores (especialmente, devido ao fato que o correlograma faz a análise do valor em relação a seus "*lagged values*").

De acordo com (WALTERS, 2018) o cálculo de médias móveis pode ser usado para identificar mais facilmente uma alta-variação (*high-variance*) nos dados de uma série temporal. E os cálculos de diferença (*diff*), variação percentual e transformação logarítmica são úteis para transformar dados não-estacionários em estacionários. Ou seja, calcular a diferença é um modo de remover tendências e tornar a série estacionária, facilitando a análise por determinados métodos. Ainda de acordo com a mesma referência, o uso de *log* pode auxiliar no tratamento de séries em que a variação (*variance*) nos dados aumenta com o tempo. O autor sugere o uso combinado de *diff* e *log* para eliminação de tendência e variação nos dados. Além disso, no caso de séries com alta volatilidade, o cálculo de médias móveis ajuda a identificar tendências.

Com isso, nos processos de engenharia de características (*feature engineering*), foram selecionados alguns dos métodos descritos acima, bem como foram realizados experimentos com métodos de predição de séries temporais muito difundidos como SES (*Simple Exponential Smoothing*) (OSTERTAGOVA; OSTERTAG, 2012) (HYNDMAN, 2014), HOLT (HOLT, 1957), ARIMA (HYNDMAN et al., 2019b) e (HYNDMAN; KHANDAKAR, 2008), entre outros.

Outro estudo realizado foi a verificação de aplicabilidade de técnicas como normalização e padronização dos dados, consideradas por alguns autores como em (SMYL; RANGANATHAN; PASQUA, 2018) que faz o uso de normalização em intervalos de tempo (janelas) como uma etapa do pré-processamento das séries:

- Normalização: A normalização ajusta a escala de valores de atributos numéricos para a faixa de 0 a 1. É um processo útil para processos influenciados pela magnitude dos valores e faz o treinamento ser menos sensível à escala de valores dos atributos.
- Padronização: Faz um deslocamento na distribuição de atributos para que tenham uma média igual a 0 e um desvio padão de 1. A padronização, tal como a normalização, também auxilia na comparação de atributos que tenham diferentes unidades ou escalas.

Os autores em (TALAGALA; HYNDMAN; ATHANASOPOULOS, 2018) sugerem que, ao invés de trabalhar com uma série temporal como uma simples sequência de observações individuais, preferem trabalhar com a análise do que chamam de "*feature space*". Destacam como características de uma série temporal (*features* selecionadas pelos autores do artigo e outras referências por eles citadas), entre outras:

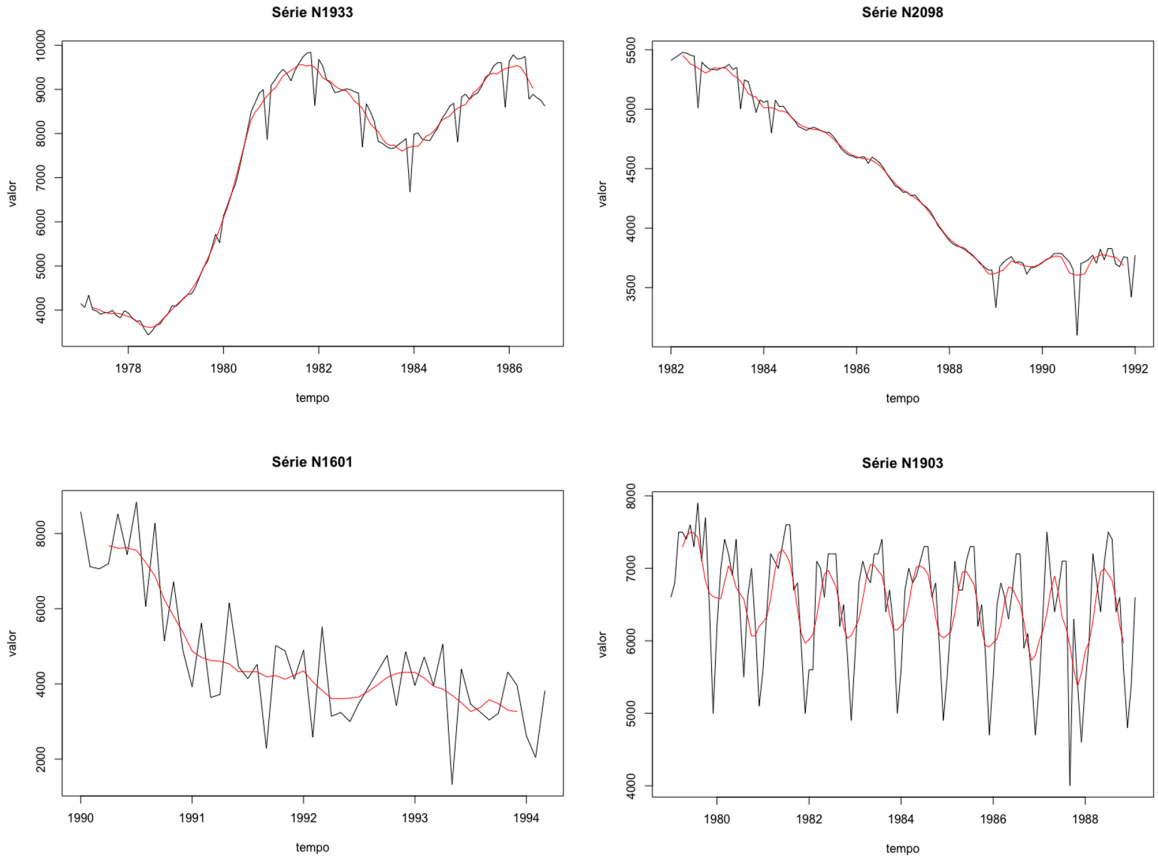
- Sazonalidade (*strength of seasonality*).
- Tendência (*strength of trend*).
- Autocorrelação.
- Heterogeneidade.
- Entropia espectral (*spectral entropy*).
- Medidas de não-linearidade e auto-similaridade.
- Comprimento (*length*) da série temporal.
- Assimetria e Curtose.
- Periodicidade.

- Medida de predicabilidade.
- O número de *turning points*.
- Mudanças de passo (*step changes*).

Analisar as características de cada série pode auxiliar no processo do método de *forecasting* a ser usado. Sobre as características das séries de M3, por exemplo, é possível visualizar na Figura 38 (com alguns exemplos de séries), que existe uma grande diversidade no perfil das séries e isso certamente tem impacto no processo de predição.

Após diferentes análises, a principal estrutura selecionada pelos experimentos para as Fases 1 e 2 deste estudo foi a estrutura denominada de "estrutura II_C", descrita na Tabela 47.

Figura 38 – Exemplos de séries temporais de M3, com observações das séries em preto e curvas suavizadas por média móvel (6 meses) em vermelho, sem normalização.



Fonte: Autoria própria

Tabela 47 – Tabela com lista de atributos utilizados na estrutura denominada de *"estrutura II_C"*

Atributo	**Tipo**	**Descrição**
Events	numeric	Valor da observação da série para o ano/mês em questão
AnoSerie	numeric	Ano da observação
MesSerie	numeric	Mês da observação
BimSerie	numeric	Bimestre da observação
TriSerie	numeric	Trimestre da observação
QuaSerie	numeric	Quadrimestre da observação
SemSerie	numeric	Semestre da observação
AR_W6	numeric	Predição calculada com auto.arima com base numa janela de 6 registros anteriores ao da observação corrente
AR_W12	numeric	Predição calculada com auto.arima com base numa janela de 12 registros anteriores ao da observação corrente
AR_W18	numeric	Predição calculada com auto.arima com base numa janela de 18 registros anteriores ao da observação corrente
AR_W24	numeric	Predição calculada com auto.arima com base numa janela de 24 registros anteriores ao da observação corrente
AR_W6Ajustado	numeric	Valor de predição de auto.arima para uma janela de 6 últimos registros ajustado a partir de cálculo de erro entre o valor predito e o total de observações para o último mês da janela
AR_W12Ajustado	numeric	Valor de predição de auto.arima para uma janela de 12 últimos registros ajustado a partir de cálculo de erro entre o valor predito e o total de observações para o último mês da janela
AR_W18Ajustado	numeric	Valor de predição de auto.arima para uma janela de 18 últimos registros ajustado a partir de cálculo de erro entre o valor predito e o total de observações para o último mês da janela
AR_W24Ajustado	numeric	Valor de predição de auto.arima para uma janela de 24 últimos registros ajustado a partir de cálculo de erro entre o valor predito e o total de observações para o último mês da janela
AR_W24_fc1	numeric	*Forecast* para horizonte de 1 mês calculado com auto.arima com base numa janela de 24 registros anteriores ao da observação corrente
AR_W24_fc2	numeric	*Forecast* para horizonte de 2 meses calculado com auto.arima com base numa janela de 24 registros anteriores ao da observação corrente
AR_W24_fc3	numeric	*Forecast* para horizonte de 3 meses calculado com auto.arima com base numa janela de 24 registros anteriores ao da observação corrente
AR_W24_fc4	numeric	*Forecast* para horizonte de 4 meses calculado com auto.arima com base numa janela de 24 registros anteriores ao da observação corrente
Lag1	numeric	Valor da observação da série para o mês anterior
shift_MA2	numeric	Média móvel calculada com base nos últimos 2 meses, defasada de 1 mês

Fonte: Autoria própria.

A.1.1 *Benchmarks*

Os autores em (AHMED et al., 2010) apresentam um estudo de predição com 1.045 séries de M3 em que utilizam algoritmos de Aprendizagem de Máquina. O estudo é citado, também, por (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018b) numa

análise comparativa do desempenho desses algoritmos em relação aos utilizados pelos competidores de M3.

Os valores obtidos e destacados como melhores por (AHMED et al., 2010) foram alcançados pelo uso de modelos MLP (*Multilayer Perceptron*), BNN (*Bayesian Neural Network*) e GP (*Gaussian Processes*), com o uso de MOV-AVG (médias móveis) no pré-processamento. Os valores de sMAPE obtidos por esses algoritmos para o horizonte de predição de 18 meses foram os seguintes:

- $sMAPE_{MLP} = 0,0834$ (8,34%)
- $sMAPE_{BNN} = 0,0858$ (8,58%)
- $sMAPE_{GP} = 0,0962$ (9,62%)

Em relação aos métodos estatísticos ETS e ARIMA, (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018b) destacam os seguintes valores (para o mesmo horizonte de 18 meses):

- $sMAPE_{ETS} = 0,0712$ (7,12%)
- $sMAPE_{ARIMA} = 0,0719$ (7,19%)

Para efeitos de *benchmarking* inicial, a Fase 1 deste estudo levou em consideração buscar, com o uso de algoritmos de mineração de fluxos de dados, resultados melhores que os obtidos por (AHMED et al., 2010) pelo uso de algoritmos de aprendizagem de máquina. Com isso, tomou-se como meta, obter valores de **sMAPE menores que 0,0834**, que corresponde ao melhor valor obtido pelo estudo tomado como referência por um algoritmo de aprendizagem de máquina.

A.2 Descrição dos Experimentos - FASE 2

Nessa fase, os experimentos foram realizados com um conjunto de 48.000 séries temporais mensais extraídas dos dados da competição M4, com o intuito de ampliar o estudo para uma base de séries maior que a utilizada na Fase 1.

De acordo com (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a), a competição M4, assim como as demais que a antecederam dessa série de *M-Competitions*, ou *Makridakis-Competitions*, têm o propósito de aprender como aprimorar a acurácia do processo de predição.

As séries mensais desse conjunto M4 representam informações extraídas de diferentes domínios (os mesmos domínios que os identificados na competição M3, como Microeconomia,

Indústria, Macroeconomia, Finanças, Dados Demográficos e Outros) e possuem as seguintes características estatísticas:

- média = 216,30 eventos por série.
- mínimo = 42 eventos por série.
- máximo = 2.794 eventos por série.
- desvio padrão = 137,41.

Em relação ao processo de *feature engineering*, pode-se dizer que, nessa Fase 2, foram utilizados os mesmos princípios e estruturas que os propostos na Fase 1 deste estudo, reproduzindo alguns dos experimentos e ajustando os métodos de processamento à medida que novos experimentos eram realizados.

A.2.1 *Benchmarks*:

Devido ao uso de um conjunto de séries diferente do utilizado na Fase 1 dos experimentos, foi necessário estabelecer novos valores de referência (*benchmarks*) a serem tomados como meta para os resultados dos experimentos realizados. Esses valores tomaram como base alguns valores selecionados dos resultados da competição M4 e o uso de um valor obtido pelo uso do algoritmo automático auto.arima.

Os valores estabelecidos como referência são os seguintes:

- sMAPE obtido pelo método 1^o colocado na competição M4:

  $sMAPE_{1stM4} = 12,126\%$

- sMAPE obtido com a utilização de auto.arima na predição das séries:

  $sMAPE_{auto.arima} = 13,491\%$

- sMAPE obtido pelo método de *Machine Learning* melhor colocado em M4:

  $sMAPE_{BestML} = 13,973\%$

A título de informação complementar, os valores obtidos pelos diferentes métodos na competição M4 para a predição de 48.000 séries mensais, podem ser visualizados na Tabela 48, que apresenta um trecho da tabela comparativa de performance de 17 métodos submetidos à competição M4 disponibilizada em (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a), mostrando apenas os valores de sMAPE para as 48.000 séries mensais. Observação: a ordem de apresentação dos métodos na tabela, corresponde à ordenação original que leva como base o desempenho de cada equipe na predição do dataset de 100.000 séries da competição.

Tabela 48 – Trecho da tabela comparativa de performance de 17 métodos submetidos à competição M4 disponibilizada em (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a).

Tipo	**Autores**	**Afiliação**	**sMAPE (séries mensais)**
Híbrido	Smyl, S.	Uber Technologies	12,126
Combinação	Montero,-Manso P.; Talagala, T.; Hyndman, R.J.; Athanasopoulos, G.	Monash University	12,639
Combinação	Pawlikowski, M.; Chorowska, A.; Yanchuk, O.	ProLogistica Soft	12,747
Combinação	Jaganathan, S.; Prakash, P.	Individual	12,487
Combinação	Fiorucci, J.A.; Louzada, F.	University of Brasilia, University of São Paulo	12,737
Combinação	Petropoulos, F.; Svetunkov, I.	University of Bath, Lancaster University	12,888
Combinação	Shaub, D.	Harvard Extension School	12,839
Estatístico	Legaki, N.Z.; Koutsouri, K.	National Technical University of Athens	13,002
Combinação	Doornik, J.; Castle, J.; Hendry, D.	University of Oxford	12,780
Combinação	Pedregal, D.J.; Trapero, J.R.; Villegas, M.A.; Madrigal, J.J.	University of Castilla-La Mancha	13,151
Estatístico	Spiliotis, E.; Assimakopoulos, V.	National Technical University of Athens	13,142
Combinação	Roubinchtein, A.	Washington State Employment Security Department	12,911
Outro	Ibrahim, M.	Georgia Institute of Technology	13,321
Combinação	Kull, M., et al.	University of Tartu	13,290
Combinação	Waheeb, W.	Universiti Tun Hussein Onn Malaysia	12,770
Estatístico	Darin, S.; Stellwagen, E.	Business Forecast Systems (Forecast Pro)	13,058
Combinação	Dantas, T.; Oliveira, F.	Pontifical Catholic University of Rio de Janeiro	13,462
Melhor ML	Trotta, B.	Individual	13,973
Segundo Melhor ML	Bontempi, G.	Université Libre de Bruxelles	14,800

Fonte: Adaptada do artigo original (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a)

A.3 Resumo dos Conceitos Estudados nos Experimentos

Esta seção apresenta um resumo com os principais conceitos estudados e atividades realizadas e conclusões obtidas nos experimentos desenvolvidos nas Fases 1 e 2.

Fase 1 - EXPERIMENTO 1.1

Nesse experimento, foram efetuados testes de predição com os algoritmos de mineração de fluxos de dados AmRulesRegressor e ARF-REG usando os dados das séries mensais de M3, com um modelo de processamento similar ao proposto por (FILHO, 2018) em que as séries estão agrupadas num único arquivo (*melted*), de forma a obter um modelo único de treinamento (um modelo global) para todas as séries.

Os arquivos ARFF foram preparados para o experimento com a adição de características (*features*) derivadas dos dados originais da série, como *lags*, *diffs* e médias móveis, e a adição de atributos derivados de algoritmos estatísticos como SES e HOLT.

Como resultado do experimento, foi obtido um erro de predição para os registros de testes no valor de sMAPE = 0,00534 (0,534%). No entanto, foi identificado que o processamento contínuo dos arquivos ARFF, gerados antecipadamente com dados de toda a série (incluindo dados de treinamento e de testes) invalidava o experimento visto que os dados de testes foram incluídos nos *datasets* utilizados no processamento. O cálculo de erro de predição (sMAPE), muito baixo em relação aos *benchmarks* estabelecidos para o conjunto de dados, ajudou a alertar sobre possível erro na abordagem utilizada no experimento.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2UXkMEK (documento externo).

Fase 1 - EXPERIMENTO 1.2

Para esse experimento, foram efetuados ajustes na estratégia de cálculo de atributos do tipo médias móveis com o objetivo de evitar o *data leakage* observado no processo de treinamento no experimento anterior. Além disso, foram adicionados novos atributos no processo de *feature engineering* na tentativa de alcançar melhores resultados no processo de aprendizagem. Para a criação de características, além dos algoritmos SES e HOLT usados anteriormente, foram utilizados os algoritmos WINTERS, ARIMA, DAMPED, RWF, SNAIVE, THETA, ETS, STLM.

Foi observado que, com as mesmas estruturas usadas no Experimento 1.1 (A, B, C, D e E) e o ajuste na forma de cálculo de médias móveis, que o melhor valor de sMAPE observado foi igual a 0,1105024 no processamento da série. No cálculo de sMAPE para os 18 registros de testes (processados porém de forma contínua com os dados de treinamento da série), o melhor resultado obtido foi 0,1209105, bem superior aos valores de *benchmark* com os quais se estava trabalhando.

Com o uso de estruturas com mais atributos foi possível obter um sMAPE igual a 0,0631299, certamente bem competitivo. Com base nisso, um teste com o uso de predição *one-step-ahead* (um passo adiante), foi realizado para o primeiro mês de teste de cada série.

Para o teste, foram selecionados os dados de todas as séries, ignorando os 18 registros finais de cada série. Além disso, um registro para predição (com atributos criados da forma já utilizada, mas deixando o campo target, *Events*, igual a 0) foi adicionado para cada série.

Depois do processamento das estruturas de dados no MOA, foi efetuado o cálculo de erro considerando apenas o primeiro registro de teste de cada série. Ou seja, o valor de sMAPE foi calculado considerando apenas 1428 registros, referentes a um registro para cada série mensal de M3. Nesse teste, o melhor valor de sMAPE obtido (para 1 registro de teste de cada série) foi igual a 0,2755060, indicando que o resultado, nessa amplitude de valor, não é competitivo e que ajustes nos experimentos deve ser realizado.

Em resumo, os dados das séries foram agrupados num único arquivo e, para as predições realizadas com o algoritmo AmRulesRegressor, foi utilizado o conceito de predição *one-step-ahead forecast*. Os experimentos com o arquivo único e geração de um modelo global de aprendizagem, mostraram resultados ainda não competitivos com os obtidos pelas técnicas usadas na competição M3.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/38xImMb (documento externo).

Fase 1 - EXPERIMENTO 1.3

A utilização de técnica de **janelas deslizantes** para a seleção de registros a serem considerados no processo de criação de características (*feature engineering*) foi experimentada nessa fase.

Como *benchmarking* para os testes, foi optado pela inclusão de valores de predição obtidos por auto.arima como mais um valor de referência para comparação com os resultados obtidos pelos métodos de mineração de fluxos de dados.

Os testes continuaram a ser feitos com as séries agrupadas num arquivo único, porém separando dados de treinamento dos dados de teste. A criação de características foi feita com a utilização dos algoritmos ARIMA, NNE, RWF, SNAIVE, THETA. Para o treinamento e predição das séries, foram usados os algoritmos de fluxos de dados AmRulesRegressor, ARF-REG, FIMT-DD, ORTO e AdaGrad.

Os valores de SMAPE observados para *one-step-ahead forecasting* considerando dados para testes ainda estão longe de uma faixa ideal, pois estão muito elevados (0,2417537, para *forecast* de 1 registro *one-step-ahead*).

Em 58,7% das séries, o menor valor de sMAPE foi obtido com o processamento das séries usando o conceito de janelas deslizantes. Essa verificação foi feita com o processamento de todas as séries usando o algoritmo autorregressivo auto.arima. Isso

evidencia que a abordagem de janelas deslizantes é promissora e merece ser explorada em outros experimentos.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2wmqGF6 (documento externo).

Fase 1 - EXPERIMENTO 1.4

Os principais conceitos explorados nesse experimento foram a inclusão de erros de predição como atributos do conjunto de dados das séries, com o objetivo de verificar se auxiliam no processo de aprendizagem, e a utilização do conceito de janelas deslizantes com janelas de 6, 12, 18 e 24 registros.

Foram efetuados estudos, também, com um subconjunto de 1.086 séries mensais de M3 (séries com total de observações de treinamento+testes superior a 80 observações) de forma a tornar o estudo mais comparável aos *benchmarks*. Essas séries foram agrupadas num arquivo único e os testes de predição foram efetuados com os algoritmos de DSM AmRulesRegressor, ARF-REG, FIMT-DD, ORTO e AdaGrad.

Analisando o menor valor de sMAPE (0,5729581) obtido pelo processamento de arquivo que incluía um (01) registro de teste de cada série (*one-step-ahead forecast*) e, depois, os valores obtidos pela análise *two-step-ahead* (sMAPE = 0,8313828) e *three-step-ahead* (sMAPE = 0,9951349), observa-se que os valores ficam distantes da *baseline* esperada para os 18 registros de testes, que deveriam se limitar a um sMAPE em torno de 0,08 para ainda se tornar competitivo.

Como resultado geral do experimento, os valores obtidos no treinamento das 1.086 séries, ao invés do total de 1.428 séries mensais de M3, não foram melhores que os obtidos no Experimento 1.3, sugerindo a necessidade de continuar os experimentos.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/39ETLtA (documento externo).

Fase 1 - EXPERIMENTO 1.5

Para esse experimento, foram efetuados estudos para verificar se o **alinhamento temporal** das séries poderia gerar melhores resultados no processo de aprendizagem.

Para testes com alinhamento temporal, as séries foram agrupadas num único arquivo e ordenadas de acordo com a data de início de cada série numa tentativa de verificar se esse tipo de organização poderia auxiliar no processamento das séries.

Foi avaliado, também, se o **processamento individual das séries** poderia ser uma abordagem melhor que o processamento conjunto das séries temporais. Além disso,

foi mantida a estratégia de análise com uso de janelas deslizantes, utilizando para os experimentos janelas de 6, 12, 18 e 24 meses.

Para o processamento das séries, além dos algoritmos de DSM utilizados no Experimento 1.4, foi incluído o algoritmo RandomAMRules. Em relação ao processo de *feature engineering*, foram adicionados atributos gerados com o algoritmo auto.arima.

O processamento individual das séries não favoreceu melhores resultados. Apenas nos domínios relativos a dados demográficos e de macroeconomia foram possíveis obter sMAPE médios inferiores a 0,08 (8%).

Em testes com alinhamento temporal foi possível obter sMAPE = 0,0921 para a base de treinamento usando o algoritmo AdaGrad. Porém nos testes de *one-step-ahead forecasting*, o sMAPE obtido foi igual a 0,9948, um resultado muito distante do *benchmark*. Então, apesar de o alinhamento ter favorecido um bom valor no processo de treinamento, a predição *one-step-ahead* foi muito prejudicada.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2OY1Odc (documento externo).

Fase 1 - EXPERIMENTO 1.6

Nesse experimento, foi efetuada uma transformação do atributo relativo à data de cada observação das séries temporais na tentativa de criar atributos que auxiliem o processo de regressão qualificando melhor cada observação em relaçâo à sua posição temporal.

Foram efetuados testes com repetição de valores (***data augmentation***) e estudos do comportamento de AMRules Regressor para definir melhor arranjo de dados.

Nas análises em que as séries são agrupadas em um único arquivo, foi efetuada a substituição do valor do atributo *Events=0* (que era o valor estabelecido para o registro a ser previsto), por outro valor estimado mais próximo ao do valor esperado para *Events* (valor da observação da série), como por exemplo: ARIMA com menor erro, ou média de valores preditos por ARIMA.

Foi efetuado, também o cálculo de sMAPE para *one-step-ahead forecast* utilizando como base as predições geradas pelo algoritmo auto.arima, com o objetivo de estabelecer nova *baseline* para comparação de valores de sMAPE.

Como conclusão do experimento, observou-se que os testes com incremento de dados (*data augmenting*) não ajudaram a ter ganhos no processo de predição.

A adição de atributos relativos à data da observação foi capaz de auxiliar de forma significativa o desempenho alcançado por algoritmos de regressão, especialmente observado na análise dos resultados obtidos por AdaGrad para *one-step-ahead forecast*.

Para o processamento com arquivos individuais por série, a ***estrutura II_C*** (com atributos como *lags*, *diffs*, médias móveis e *features* geradas com auto.arima) e o uso do algoritmo AdaGrad foi possível obter o menor valor de sMAPE (0,0980) em comparação aos demais algoritmos e abordagens que vinham sendo utilizadas nos experimentos.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/38DQ87q (documento externo).

Fase 1 - EXPERIMENTO 1.7

Nessa fase, foram continuados os estudos com o algoritmo AdaGrad por ter apresentado melhores resultados em experimentos anteriores. Além disso, foram efetuados estudos comparativos com o método ***Prophet*** e auto-arima.

Experimentos com o conceito de ***cluster*** foram efetuados para tentar identificar algum parâmetro de agrupamento útil para a análise das séries.

Foram continuados os testes com análise de séries conjuntas, incluindo atributos de variação que possam ser incrementais, sem incluir informações sobre a categoria da série na análise.

Os métodos de DSM utilizados nesse experimento foram os seguintes: AmRulesRegressor, ARF-REG, FIMTDD, ORTO, ADAGRAD, FADING TARGET MEAN, RandomRules e RandomAMRules.

Em comparação entre Prophet, auto.arima e AdaGrad, o algoritmo auto.arima (ARIMA) apresentou melhores resultados gerais. No entanto, em 64% das séries da categoria Microeconomia, Prophet apresentou resultado melhor que auto.arima e, no geral, Prophet obteve melhores resultados em 374 séries. AdaGrad conseguiu melhores resultados que auto.arima em 436 séries, com melhor desempenho também na série Microeconomia em que foi melhor em 51% das séries. Com isso, foi possível observar que, apesar de auto.arima apresentar melhor resultado geral no processo de predição, os outros dois algoritmos conseguem superar os valores de auto.arima em determinadas séries. Isso sugere que **o uso conjunto de diferentes abordagens pode ser uma opção com melhor resultado final** ao realizar o *forecasting* para múltiplas séries.

O estudo do uso de técnicas de *clustering* (agrupamento) com o objetivo de agrupar séries por similaridade em suas características (perfil), mostrou que é possível identificar séries que se agrupam, porém não evidencia de forma clara os atributos a serem utilizados para a seleção do algoritmo de predição ideal para cada série.

Em testes com séries agrupadas e otimização de parâmetros foi possível obter melhoria de desempenho com o algoritmo AdaGrad, chegando a um sMAPE para *one-step-ahead forecast* = 0,0962 (no Experimento 1.6, o melhor resultado tinha sido 0,0980).

Em relação à otimização de parâmetros, foi possível observar que AdaGrad é bem sensível a variação no parâmetro *LearningRate*, correspondente à taxa de aprendizagem. Para RandomRules, com a variação nos parâmetros *numAttributesPercentage* e *EnsembleSize* foi possível observar variação na acurácia do algoritmo.

A adição de 39 características extraídas das séries (como atributos relativos a tendência, sazonalidade, linearidade, entropia, curvatura, entre outros) com o uso de funções do pacote *tsfeatures* (HYNDMAN et al., 2019), chegando a um *dataset* com 105 *features*, não apresentou resultados muito melhores que os obtidos anteriormente. Isso demonstra que os algoritmos de mineração de fluxos de dados se beneficiam de atributos que possam expressar melhor as características específicas de cada série e não necessariamente por um grande número de atributos que, apesar de serem atributos extraídos das próprias séries, não sejam capazes de favorecer a discriminação das séries entre si.

A tentativa de eliminar dos *datasets* das séries os atributos derivados de auto.arima mostrou, por sua vez, que os melhores resultados são obtidos ao manter esses atributos nos dados das séries.

O sMAPE médio (mínimos entre Prophet e Adagrad) = 0,0626 (obtido pela seleção manual das séries) sugere que, com a combinação entre algoritmo de mineração de fluxo de dados e algoritmo *batch*, é possível alcançar melhores resultados no processo de predição.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/323vRFQ (documento externo).

Fase 1 - EXPERIMENTO 1.8

Nessa fase, foram efetuadas análises com as 1.045 séries mensais de M3 utilizadas por (AHMED et al., 2010) de forma a tornar o experimento mais comparável ao estudo de Ahmed.

Ao efetuar cálculos de *forecasts* Mínimos, Máximos e Médios para os algoritmos auto.arima, Prophet, AdaGrad e Random Rules e analisar os valores de sMAPE alcançados pela combinação desses diferentes métodos, foi confirmado que a combinação é capaz de gerar resultados melhores que os obtidos individualmente por AdaGrad. No entanto, os valores de sMAPE obtidos pelo *ensemble* ainda não são competitivos com os valores obtidos isoladamente pelo uso de auto.arima, tanto para 1.045 quanto para a análise de 1.086 séries. Para a combinação de AdaGrad, auto.arima e Prophet, foram obtidos os seguintes valores de sMAPE médio para o horizonte de h1 a h2: para 1045 séries, sMAPE = 0,1052; para 1086 séries, sMAPE = 0,1054.

Novos estudos de combinações de algoritmos são necessários para continuar na busca por melhores resultados, justificando, assim, a continuação dos experimentos.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/37A5jgm (documento externo).

Fase 1 - EXPERIMENTO 1.9

Nesse experimento, foi avaliada a utilização de processo de **seleção de algoritmo** com base em desempenho em etapas de treinamento e validação.

Foram considerados no estudo, grupos de 1.045 e de 1.086 séries mensais de M3 descritos em fases anteriores deste estudo.

Em relação aos testes com os algoritmos de DSM, foi utilizado o AdaGrad para os testes com arquivos individuais por séries e o algoritmo RandomRules para os testes com séries agrupadas num único arquivo.

Foram utilizadas duas abordagens para a análise de *ensemble* para séries processadas individualmente: na abordagem A, a seleção foi efetuada com base em cálculo de sMAPE em *subset* de validação (base de treinamento separada de base de validação em proporção 80/20); e, na abordagem B, a seleção foi efetuada com base em valores de treinamento.

A abordagem que apresentou menor sMAPE para o *forecast* de horizonte de um mês (H1) foi a que faz a seleção de algoritmos com base no sMAPE resultante da análise de 40% dos registros de treinamento. Essa abordagem utiliza predições geradas pelo MOA em modo *prequential* como base para o algoritmo AdaGrad e dados de *fitting* (treinamento) gerados pelo auto.arima para a série.

Para 1.086 séries, o sMAPE obtido pela combinação de algoritmos para H1 foi igual a 0,0800. Para 1.045 séries, o sMAPE obtido pela combinação de algoritmos para H1 foi igual a 0,0804. Apesar de os valores obtidos estarem bem próximos dos valores alcançados por auto.arima quando usado de forma isolada (0,0798 para 1.086 séries e 0,0802 para 1.045 séries), para horizontes mais distantes, como o H18 (h = 18 meses de predição), os valores de sMAPE obtidos pela combinação de algoritmos fica muito superior aos obtidos por auto.arima, visto que, enquanto auto.arima apresenta sMAPE para H18=0,1202, os demais métodos apresentam valores de sMAPE superiores a 0,1535 para esse mesmo horizonte, significando que um processo alternativo deve ser buscado.

Pelos resultados obtidos para o teste com base agrupada (*melted*) e RandomRules, é possível identificar que o baixo valor de sMAPE obtido nos registros de treinamento não necessariamente guia o processo de seleção ao melhor valor possível na etapa de predição.

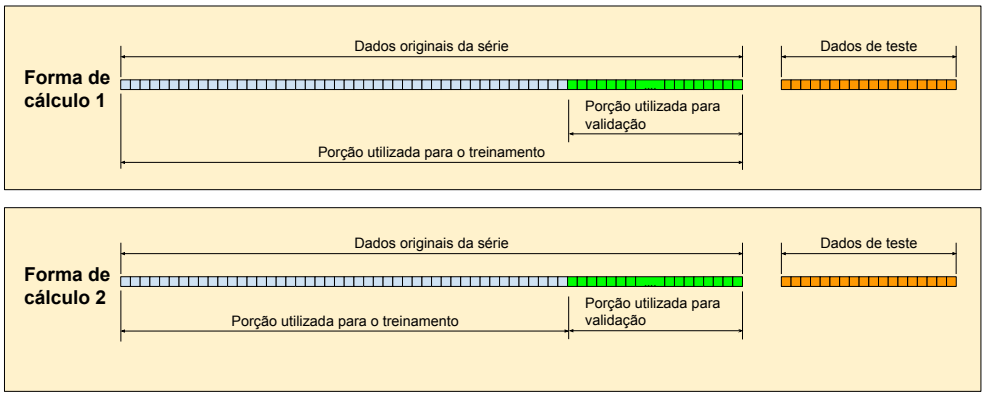
Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/320kuOX (documento externo).

Fase 1 - EXPERIMENTO 1.10

Nesse experimento, foram realizados estudos para implementação de processo de seleção de algoritmo com base em sMAPE de validação obtidos a partir de dados do treinamento das séries, calculados sob diferentes critérios. O sMAPE de validação, nesse experimento, corresponde ao erro calculado com base nos valores obtidos no treinamento e os valores reais para o mesmo intervalo da série.

A Figura 39 apresenta duas formas distintas de separação de dados originais da série (dados de treinamento) para o cálculo de sMAPE de validação. Na maioria dos experimentos das Fases 1 e 2 optou-se pelo uso da Forma de cálculo 1, fazendo o cálculo de sMAPE de validação a partir da seleção de uma parte final dos dados de treinamento (*fitting*) gerados pelos algoritmos para toda a série. Especialmente no Experimento 2.8 a Forma de cálculo 2 (que separa os dados originais da série em dados de treinamento e validação, reservando a porção final para validação) foi avaliada. Em relação à quantidade de observações utilizadas para o cálculo de sMAPE de validação, diferentes quantidades de registros foram avaliadas e estão descritas nos experimentos. Quanto ao sMAPE de teste, os experimentos foram realizados utilizando, nas análises iniciais, o horizonte de 1 mês de predição e, posteriormente, os cálculos foram realizados para todo o horizonte de predição previsto para o *dataset* utilizado, que disponibiliza 18 registros de testes para cada uma das séries.

Figura 39 – Diagrama com formas de separação de dados de treinamento para o cálculo de sMAPE de validação.



Fonte: Autoria própria

Foram consideradas no estudo, grupos de 1.045 e de 1.086 séries mensais de M3 descritos em fases anteriores deste estudo (ver Experimento 1.9).

No processamento individual de 1045 séries de M3 com o uso dos algoritmos auto.arima e AdaGrad e um critério de seleção de algoritmo baseado no valor de sMAPE

calculado com base em 40% dos dados de treinamento foi possível obter o menor valor de sMAPE para horizonte de 18 meses, obtendo sMAPE = 0,11828.

O mesmo ocorreu para os testes com processamento individual de 1086 séries de M3, onde o menor valor de sMAPE = 0,11837 foi obtido tomando como base os 40% finais dos dados de treinamento da série.

Testes realizados com as séries agrupadas não apresentaram resultados melhores que os obtidos pelo processamento individual das séries.

Pelos experimentos realizados, observa-se que existe uma diferença estatística significativa entre os valores obtidos pelo uso de auto.arima e o uso combinado de auto.arima e AdaGrad quando considerados para a seleção os valores de sMAPE calculados nos 40% dos registros finais de treinamento.

Isso sugere que, com a utilização de método de seleção que considere, de forma combinada, o uso de algoritmos estatísticos e de *machine learning* (no caso, algoritmos de *data stream mining*), podem ser obtidos resultados melhores de *forecast* para horizontes mais distantes que o uso isolado de apenas uma das técnicas. Essa percepção ficou evidenciada especialmente para o caso de séries processadas individualmente. No caso de séries processadas num único arquivo, o método de seleção foi capaz de selecionar apenas 2 séries processadas com algoritmo de *data stream mining* e as demais séries selecionadas no *ensemble* foram processadas com auto.arima.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2USLMoN (documento externo).

Fase 2 - EXPERIMENTO 2.1

A partir dessa fase, os experimentos foram realizados com 48.000 séries mensais de M4.

Nesse experimento 2.1, foi utilizada a seleção dos algoritmos auto.arima e AdaGrad, usando como base para seleção o valor de sMAPE calculado com base em 40% dos registros finais dos dados de treinamento de cada série.

O valor obtido com a seleção (sMAPE = 13,635%) como média para os 18 meses de testes não supera o valor calculado com o uso de auto.arima (sMAPE = 13,491%), mas consegue ser melhor que os valores apresentados pelos competidores de M4 que apresentaram melhores resultados usando exclusivamente algoritmos de *Machine Learning* ($sMAPE_{BestML} = 13,973\%$ e $sMAPE_{2ndBestML} = 14,800\%$).

Para horizontes mais próximos (1 a 3 meses, *short-term forecasting*), o uso do *ensemble* conseguiu resultados melhores que os apresentados pelo uso isolado de auto.arima.

Os resultados apresentados, no entanto, ainda estão muito distantes dos valores

obtidos pelos competidores com melhores resultados na competição M4, justificando avaliar outras possibilidades no uso combinado das técnicas com o objetivo de aperfeiçoar os resultados obtidos.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2SzmL0q (documento externo).

Fase 2 - EXPERIMENTO 2.2

Nessa fase, foram realizados testes de seleção tomando como base valores de sMAPE calculados a partir de diferentes percentuais de dados de treinamento (40%, 30%, 20%, 10%). Além disso, foram efetuados testes com ***data augmentation*** e AdaGrad e estudadas as técnicas utilizadas por competidores com melhores colocações em M4.

Foi possível, também, fazer a definição do oráculo a partir de predições calculadas por auto.arima e AdaGrad para o horizonte de 1 mês (H1). O oráculo, nesse caso, estabelece a combinação ideal entre séries atribuídas a AdaGrad e auto.arima que minimiza o erro (sMAPE) alcançado para as 48.000 séries em análise.

Em relação aos testes de seleção com cálculo de sMAPE com 40%, 30%, 20% e 10% dos dados de treinamento, pode-se observar que, para o horizonte de 18 meses, o melhor resultado para o *ensemble* é obtido com o uso dos 40% dos registros finais de treinamento.

Considerando os resultados apresentados pelos competidores de M4, observa-se que o método proposto já é capaz de superar os resultados obtidos pelos métodos que usam exclusivamente técnicas de *Machine Learning*. Um *benchmarking* importante a ser considerado, a partir dessa fase, pode ser o valor apresentado pelo melhor colocado na competição que foi capaz de obter um sMAPE igual a 0,12126 (12,126%) para a predição de 18 meses para as séries mensais de M4.

Em relação ao oráculo identificado para o *ensemble* e horizonte de 1 mês, deve-se buscar o método de seleção que consiga selecionar as 18.497 séries para as quais o erro para AdaGrad é menor, por parecer ser esse o cenário que representa a parcela ideal de séries a serem atribuídas a esse algoritmo.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2OUIM7u (documento externo).

Fase 2 - EXPERIMENTO 2.3

Nos experimentos dessa fase, foram realizados diferentes estudos como a análise de valores de sMAPE para o horizonte de H1 a H18 em que AdaGrad apresentou melhor desempenho, testes de seleção tomando como base 5, 12 e 18 últimos registros de treinamento e experimentos com técnicas de extração de características de séries temporais

usando funções do pacote *tsfeatures* do R.

Foram realizados, também, experimentos com o uso de **Análise de Componentes Principais (PCA)** com o objetivo de identificar *features* com maior correlação entre si de forma a verificar como melhor separar as características capazes de selecionar as séries melhor processadas com AdaGrad. Para essa análise, foram usados métodos dos pacotes *FactoMineR* (LÊ; JOSSE; HUSSON, 2008) e *factoextra* (KASSAMBARA; MUNDT, 2017) do R.

As séries selecionadas para cada método (AdaGrad e auto.arima) nos diagramas montados com base na análise PCA (Análise de Componentes Principais) se mostraram dispersas por todo o gráfico, dificultando a identificação clara da participação das variáveis na definição do critério de seleção a utilizar.

Em relação à definição de um oráculo para o *dataset* dessas séries temporais, no Experimento 2.2 havia sido definido um oráculo considerando o horizonte de apenas um mês de predição. Nesse Experimento 2.3, por sua vez, foi estabelecido o oráculo para o horizonte de 18 meses (que corresponde a todos os registros de testes). Para esse cenário, o objetivo passou a ser o de conseguir selecionar, com base nos dados de treinamento, as 15.320 séries melhor representadas pelo algoritmo AdaGrad.

Os resultados do experimento sugerem, então, ampliar os estudos em busca de melhores critérios de seleção de forma a obter um valor de sMAPE realmente competitivo com os métodos estado-da-arte. O sMAPE para o oráculo, calculado considerando os valores mínimos de sMAPE de teste obtidos com os *forecasts* de auto.arima (32.680 séries) e AdaGrad (15.320 séries), passa a ser igual a 0,11687 (11,687%).

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2u6QvIy (documento externo).

Fase 2 - EXPERIMENTO 2.4

Nesse experimento, foi efetuada a análise comparativa entre as características extraídas dos dados originais das séries e as características extraídas a partir dos dados de treinamento gerados pelos algoritmos AdaGrad e auto.arima. A comparação foi efetuada com base em todos os dados da série (série completa) e também com base nos últimos 36 registros de cada série, usando como base para a seleção, erros calculados com a métrica RMSE (*Root Mean Squared Error*).

Foram efetuadas, também, análises comparativas dos métodos tomando como base diferentes combinações de *features* extraídas das séries temporais, como por exemplo: *features* com maior correlação com dimensão principal (*Dim1*, da análise de componentes principais realizada no Experimento 2.3), entropia, heterogeneidade, entre outros, com o objetivo de experimentar outro método de seleção.

Uma análise com o **algoritmo de otimização (*xgboost*)** foi realizada na tentativa de melhorar o processo de seleção e identificar séries mais relevantes para a escolha de AdaGrad com base nas características extraídas das séries. Além disso, foram realizadas estatísticas gerais sobre as 15.320 séries em que AdaGrad se mostrou melhor em relação ao *forecast* para os horizontes do mês 1 ao mês 18 (H1-H18).

O uso de outra métrica para o cálculo de erros de predição, no caso o RMSE (*Root Mean Squared Error*) em conjunto com a análise de sMAPE de validação com base nos dados da série completa e dos últimos 36 registros de cada série (com o objetivo de avaliar os dados relativos aos últimos 3 anos de cada série) não foi capaz de apresentar resultados melhores que os que vinham sendo obtidos com os cálculos de sMAPE a partir de 40% dos registros finais de treinamento.

Experiências baseadas em diferentes combinações de características extraídas das séries com o pacote *tsfeatures* também não conseguiram gerar resultados melhores que os obtidos em experimentos anteriores.

Testes com o uso de algoritmo de otimização do pacote *xgboost* foram capazes de recomendar o uso de AdaGrad para 17.602 séries, sendo que apenas 5.693 dessas séries pertencem ao grupo das 15.320 séries do oráculo em que AdaGrad apresenta melhores resultados. Essa seleção de séries não pertencentes ao oráculo justifica o fato de os resultados obtidos com essa técnica não terem sido melhores que os obtidos em outros experimentos, visto que o sMAPE médio obtido para h1 a h18 nesse experimento foi igual a 15,504%.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/37A9LM6 (documento externo).

Fase 2 - EXPERIMENTO 2.5

Nesse experimento foi utilizado um **Algoritmo Genético** com o intuito de identificar a melhor combinação de atributos a serem utilizados nos processos de classificação. Um algoritmo implementado em Python foi utilizado para identificar as *features* com maior influência no processo de classificação. Na função-objetivo (eval_func), um algoritmo de Árvore de Decisão foi usado para avaliar uma porção do *dataset* definida para o treinamento e outra para validação (na proporção aproximada de 50% treinamento, 20% validação, 30% testes). Uma população de 10 indivíduos e 5 gerações foram utilizadas para o processamento e otimização/evolução de resultados.

Além disso, foi efetuada análise com o intuito de identificar atributos capazes de destacar as *features* mais apropriadas para a definição da classe (*target*) ideal de cada série. A análise teve caráter especial de separar as séries melhor resolvidas pelo algoritmo AdaGrad e incluiu a montagem de diferentes estruturas de dados e avaliação desses *datasets* com algoritmos de *machine learning* a fim de verificar o desempenho dos mesmos

na separação de séries a serem processadas com AdaGrad e com auto.arima.

A adição de *features* baseadas em contagens, médias e diferenças de valores referentes a dados obtidos na validação em conjunto com características extraídas das séries temporais, assim como o processamento desses *datasets* com algoritmos de *machine learning*, não se mostraram suficientes para permitir a separação precisa (ou, pelo menos, com nível de acurácia elevado) das séries melhor processadas com AdaGrad ou auto.arima.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2vCCQt0 (documento externo).

Fase 2 - EXPERIMENTO 2.6

Nesse experimento foram realizadas análises ignorando valores de treinamento referentes aos dados gerados pelo processamento da série completa, pois foi percebido que esses valores são muito distantes dos apresentados por auto.arima e poderiam estar influenciando os resultados gerados pela análise em busca do oráculo.

Foram efetuados, também, testes com um conjunto de 1.000 séries de M4 selecionadas aleatoriamente e processadas com AdaGrad usando apenas características básicas da série (como mês e total de observações) numa tentativa de verificar se os valores gerados por esse algoritmo nessa situação poderiam gerar melhor valores comparativos com os dados gerados por auto.arima.

Outros experimentos incluíram testes com *datasets* separados por categoria das séries (campo *tipo* da base de dados do pacote *M4comp2018*) a fim de identificar as categorias em que os algoritmos de classificação conseguem melhor identificar a correlação entre as propriedades das séries e as indicações dadas pelo oráculo.

Os resultados obtidos nesse experimento sugerem que a técnica de tentar identificar as séries de AdaGrad do oráculo, tomando como base os valores de validação para diferentes combinações de situações em que os erros de validação de AdaGrad são menores que os gerados por auto.arima, não demonstrou, até a conclusão desse experimento, ser um instrumento eficiente.

No entanto, pode-se observar que os resultados obtidos com o modelo de classificação definido para as séries da categoria *Outros*, que é viável continuar a buscar características melhores de discriminação das séries, em busca de um resultado melhor que o apresentado por auto.arima isoladamente. O experimento sugere, então, concentrar esforços na análise das características dessas séries de forma a identificar quais as que apresentam maior influência na definição do método ideal de processamento de cada série.

A dificuldade de padronização dos dados pela natureza do processamento *online* proposto pelo *data stream mining* dificulta estabelecer parâmetros de comparação entre

as diferentes *features* dos *datasets* e as séries analisadas. Métodos existentes na literatura sugerem o uso de padronização dos dados e o processamento *batch* de, pelo menos, parte do *dataset*. É preciso, então, estabelecer estratégias que consigam assegurar melhor acurácia no processo de predição, desconsiderando a utilização de técnicas como a de padronização dos dados.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2SPE24k (documento externo).

Fase 2 - EXPERIMENTO 2.7

Nessa fase, foi efetuada uma análise estatística do *dataset* de 48.000 séries mensais de M4, com atributos derivados a partir dos erros (sMAPE) calculados para 5, 12 e 18 regs e para 10, 20, 30 e 40 porcento dos registros de treinamento e dados das séries completas, bem como análise das *features* extraídas das séries.

Foi efetuada análise do uso da abordagem de **Fusão** considerando os valores de predição gerados por AdaGrad e auto.arima, atribuindo diferentes combinações de pesos na composição do *forecast* final de cada série. Além disso, foi efetuado estudo inicial do código-fonte disponibilizado por Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, Thiyanga S Talagala (MONTERO-MANSO et al., 2018a), autores da abordagem classificada em 2^o lugar no *ranking* da competição M4, que propõe o uso de modelo de *metalearning* baseado em features extraídas das séries e erros apresentados por 9 diferentes algoritmos estatísticos.

O estudo estatístico de atributos do *dataset* de 48.000 séries usando valores obtidos durante o treinamento (erros para diferentes combinações de registros) e *features* das séries, ainda não foi conclusivo para descartar a viabilidade de se utilizar informações dessa etapa do processo de aprendizagem de máquina (dados da fase de treinamento) como indicador do método mais adequado a ser usado para cada série.

O uso de técnica de fusão foi importante para comprovar que o uso combinado de técnicas distintas como o auto.arima (estatístico) e o AdaGrad (*data stream mining algorithm*) podem resultar em maior acurácia no processo de predição em séries temporais como as disponibilizadas no *dataset* de M4.

A abordagem proposta pelos autores em (MONTERO-MANSO et al., 2018a), em especial no que se refere à extração de hiperparâmetros baseados nas características extraídas das séries, sugere ser uma abordagem promissora como auxílio no processo de seleção do método de predição mais adequado para cada série. Estudos mais detalhados da técnica e definição de um método combinado devem ser feitos a fim de continuar a verificação sobre a possibilidade do uso de dados de treinamento como base para a seleção de método mais adequado para cada série.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/321dwJa (documento externo).

Fase 2 - EXPERIMENTO 2.8

Nesse experimento, foram realizados estudos com a **separação de séries originais em registros para treinamento e registros para validação**, para verificar se o sMAPE obtido nessa camada de validação poderia dar pistas sobre o sMAPE resultante na etapa de testes.

Trata-se de um teste diferente do realizado em etapas anteriores do experimento, em que os valores de validação eram calculados com base nos valores gerados durante a fase de treinamento (*fitting*), usando todos os registros da série para esse processo e fazendo experimentos com sMAPE de validação calculados sob diferentes condições.

Nesse experimento foram realizadas análises com as 48.000 séries de M4 e também com 2 grupos distintos de séries desse *dataset* selecionadas aleatoriamente, sendo um com 500 e outro com 5.000 séries.

Com base nos experimentos realizados, pode-se considerar que a adição de uma etapa de validação (separação dos dados da série em treinamento e validação) para os processamentos de séries com o algoritmo AdaGrad não conseguiu resultar numa melhor discriminação entre as séries melhor processadas pelo algoritmo. Apesar de se obter melhores resultados de sMAPE de testes para AdaGrad em alguns horizontes (evidenciados nos testes com 500 séries), a melhora não foi expressiva.

Isso sugere que novas abordagens de seleção devem ser exploradas de forma a conseguir identificar de forma mais assertiva as séries atribuídas a AdaGrad.

A abordagem de separação dos dados da série em dados de *Treinamento+Validação* não estava sendo feita por se acreditar que a análise contínua (*Test-Then-Train*) proposta pelos algoritmos de *Data Stream Mining*, oferece uma alternativa a esse modelo. Mas, deve-se considerar que, para esse cenário, o uso de uma abordagem mais clássica, pode trazer ganhos ao experimento como um todo.

Como próxima etapa do estudo, sugere-se continuar a busca por estratégias que possam identificar, com base nas características das séries, ou com base em outros critérios de validação, o algoritmo mais recomendado para cada série.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/2HpxzaS (documento externo).

Fase 2 - EXPERIMENTO 2.9

Nessa fase foram realizados estudos para a integração de AdaGrad na lista de algoritmos utilizados pelo método **M4metalearning** (MONTERO-MANSO et al., 2018b).

Os experimentos foram realizados com o processamento de 999 séries mensais de M4 utilizando o método M4metalearning com o algoritmo AdaGrad integrado.

Ao efetuar uma análise combinada de M4metalearning e os valores de sMAPE gerados pelo uso individual de AdaGrad, foi possível observar grande potencial de diminuição de erro. Isso sugere que uma melhor avaliação sobre a forma como M4metalearning atribui pesos a cada algoritmo pode ser objeto de estudo futuro.

Considerando que os resultados obtidos isoladamente por auto.arima foram muito próximos aos obtidos pelo uso de M4metalearning com AdaGrad, pode-se avaliar a utilização da técnica de atribuição de pesos dos algoritmos usada por esse método, considerando apenas auto.arima e AdaGrad, em especial porque os valores de sMAPE de testes obtidos individualmente por AdaGrad (com sMAPE de teste menor em 25,7% das séries avaliadas) podem melhorar muito os resultados gerais do processo se for atribuído o devido peso a esse algoritmo.

Como estudos futuros, sugere-se avaliar o uso combinado de AdaGrad, auto.arima e M4metalearning como integrantes do método de predição proposto pelo estudo.

Mais detalhes sobre esse experimento podem ser obtidos no *link* http://bit.ly/37yBGw2 (documento externo).

APÊNDICE B – Estudos para a Definição de Parâmetros para o Processo de *Feature Engineering*

Descrição Geral

Esta seção descreve os estudos e experimentos realizados para a formalização de parâmetros para o processo de *feature engineering* adotado para o método de predição AA-ACF proposto por este estudo.

Nos experimentos iniciais realizados para a estruturação do método e apresentados no Apêndice A, o processo de *feature engineering* se baseava, principalmente, na criação de atributos derivados de forecasts de auto.arima calculados para diferentes tamanhos de janelas de dados das séries. Tratava-se, portanto, de um processo muito dependente de um algoritmo com um custo computacional expressivo, visto que para cada atributo os parâmetros de auto.arima precisavam ser recalculados.

Além disso, a abordagem inicial do método avaliava o uso de *M4metalearning* paralelamente ao uso de AdaGrad e auto.arima como um método participante do *ensemble*. Porém, como o *M4metalearning* pressupõe a criação em *batch* de um modelo para estabelecer os pesos de cada algoritmo na composição do *forecast* e possui um processamento com tempo muito elevado (por envolver 9 diferentes algoritmos no processo de treinamento), alternativas mais eficientes e mais adaptáveis ao processamento de fluxos de dados foram avaliadas.

Os estudos para o processo de *feature engineering* descritos nesta seção foram baseados na análise do conceito de dependência temporal de forma a introduzir nos atributos da série informações históricas identificadas nos próprios eventos da série. Além de estabelecer a estrutura geral para o processo, os estudos também foram responsáveis pela seleção dos parâmetros mais adequados a serem utilizados no processamento de séries de diferentes naturezas, mais especificamente para a definição de tamanho de janela deslizante a ser utilizada para a geração de coeficientes de autocorrelação e número de coeficientes de autocorrelação a serem considerados na criação de atributos.

B.1 Resumo dos Conceitos Estudados nos Experimentos

A seguir são descritos os estudos e experimentos realizados na etapa identificada como Fase 3 do estudo.

Fase 3 - EXPERIMENTO 3.1

Nesse experimento foram realizados estudos iniciais sobre o tema Dependência Temporal. O estudo foca na abordagem denominada de *Temporally Augmented Classifier* citada pelos autores em (ŽLIOBAITė et al., 2015) que a descrevem como uma técnica que se baseia em adequações no processo de pré-processamento e não requer modificações no modelo preditivo, permitindo que qualquer classificador seja usado como base com essa estratégia. A técnica sugere o treino do classificador com o uso de vetores de entrada aumentados (*augmented input vectors*) pela adição de informações nos dados de entrada. Tendo esse conceito como referência, a ideia de avaliar o uso de coeficientes de autocorrelação (ACF, *autocorrelation function*) passou a ser explorada em mais profundidade a partir desse ponto.

Em experimentos realizados com *features* criadas com base em ACF foi possível observar que a adição de atributos derivados de coeficientes de autocorrelação da série, normalizados e ajustados para uma escala dentro da faixa de valores das observações da série, apresentam melhores resultados que com o uso dos coeficientes sem adequação de escala. Experimentos com cálculo de coeficientes de ACF com janelas de 72 registros (correspondentes a 6 vezes a frequência das séries mensais, estabelecida como 12 observações), apresentam resultados melhores (menor valor de sMAPE de testes) que os obtidos com janelas de 36 observações. Com a ampliação do tamanho da janela para 144 registros e consideração de 18 coeficientes ACF, foi possível obter um ganho de 26,7% em relação aos testes efetuados com janela de 72 registros.

Ainda em relação ao uso de dependência temporal, outra abordagem avaliada neste estudo foi o uso de filtro de Kalman, também reconhecido como aplicável para a área de predição de séries temporais por levar em conta aspectos relativos a dependência entre diferentes momentos (estados) de uma série e por sua implementação que sugere similaridade a um modelo oculto de Markov. Experimentos realizados com o uso de filtro de Kalman e técnica de Modelo Linear Dinâmico no processamento de séries temporais mostram que a técnica é competitiva com valores gerados pelo uso isolado de auto.arima. No entanto, os testes efetuados com o uso de filtro de Kalman para a criação de *features* para *datasets* processados com AdaGrad não foram capazes de confirmar a competitividade desse filtro. Nota: como os testes foram feitos de forma a tentar usar o filtro de Kalman e (de forma conjunta) uma estrutura com menos atributos, pode ser que a tentativa de explorar essas diferenças simultaneamente comprometeram a qualidade da análise,

sugerindo que outros testes poderiam ser realizados nas mesmas condições que as utilizadas para o processamento com AdaGrad com *features* de auto.arima. Além do aumento nos valores de erro, outra percepção que se teve foi a de aumento no tempo de processamento para a geração de *features*, mesmo num cenário com apenas um atributo derivado de filtro de Kalman.

Além dos estudos sobre dependência temporal, nessa etapa foram efetuados estudos com variações na taxa de aprendizagem de AdaGrad fazendo simulações com as taxas 0,1 e 0,01 (default). Também foram efetuados também alguns testes com o uso de Adadelta, outro algoritmo de DSM baseado na técnica de descida do gradiente, a fim de compará-lo ao desempenho obtido com AdaGrad.

Em testes realizados com AdaGrad num conjunto de 1000 séries do *dataset* M4, o uso de taxa de aprendizagem = 0,1 (maior que o *default* que é igual a 0,01) não foi capaz de apresentar melhora nos resultados gerais, mesmo para as séries com total de observações menor ou igual a 100. O resultado observado num conjunto menor de séries (que sugeriam possibilidade de melhores resultados com a alteração de *learning rate*) não se repetiu nas séries do experimento.

Testes com os algoritmos AdaGrad e Adadelta do pacote gradDescent (WIJAYA et al., 2018) para o R, mostraram que Adadelta apresentou melhor resultado para séries de até 100 observações. Além desse valor, os processamentos efetuados com AdaGrad apresentaram menor sMAPE no teste de *one-step-ahead forecast* (para o teste para horizonte de predição de 1 mês). Foram avaliadas fontes explicando o comportamento do algoritmo e principais diferenças em relação ao AdaGrad, porém não foram feitos testes com adaptação funcional do algoritmo para execução no MOA.

Os resultados obtidos com o uso de *features* calculadas com base em valores de coeficientes de autocorrelação sugerem que a criação de atributos com o uso de técnicas que levam em conta o conceito de dependência temporal podem auxiliar no desempenho de algoritmos de DSM (como o AdaGrad). Apesar de os resultados dos testes iniciais não terem superados os valores obtidos pelo uso isolado de auto.arima, foi possível observar que alterações simples (como no tamanho da janela deslizante a ser considerada na geração de coeficientes) podem gerar melhoras expressivas no comportamento de AdaGrad mesmo com o uso de atributos facilmente calculados (e que demandam menos esforço computacional que os atributos derivados de auto.arima que vinham sendo utilizados).

Com base nisso, sugere-se avaliar a adição de novos critérios na geração das *features*, como o uso de coeficientes de autocorrelação parcial (PACF, *Partial autocorrelation function*) e ajustes no processo de normalização na de escala de valores.

Nota: por se tratar de um estudo de predição de séries temporais univariadas, a principal dificuldade está no estabelecimento das *features* que vão compor o *dataset*

submetido ao processo de treinamento. A dificuldade fica maior ainda quando se demanda predições para horizontes mais distantes (como o intervalo de 1 a 18 meses avaliado nesse estudo), uma vez que, como as *features* dos registros de testes são compostas por informações artificiais (ou informações passadas das séries), o erro acumulado no processo iterativo de geração das predições torna o processo sujeito a apresentar maior erro nas predições para meses mais distantes. Sobre essa questão, o estabelecimento do valor do atributo *target* (classe a ser predita), inclusive, pode interferir no rendimento do processo, especialmente porque o modelo gerado pelo algoritmo de DSM é sujeito a sofrer atualização a cada predição pela natureza do modelo *test-then-train* que, ao confrontar o valor previsto com o valor estimado submetido para a predição, procura se ajustar ao erro percebido durante esse processo. Por isso, um esforço combinado de geração de *features* e de definição de valor para atributo *target* deve ser realizado.

Mais detalhes sobre esse experimento podem ser obtidos no *link* https://bit.ly/2TKLT3K (documento externo).

Fase 3 - EXPERIMENTO 3.2

Nessa fase foram realizados novos experimentos com *features* criadas com base em coeficientes de autocorrelação, denominadas de *features* de ACF. Foram efetuados testes com janelas deslizantes de 144 registros e ajustes na forma de normalização dos coeficientes. Adicionalmente, foram experimentadas também janelas deslizantes de 288 e 480 registros, procurando trabalhar com valores múltiplos dos valores mais comuns de frequência das séries. Variações nos números de coeficientes de autocorrelação também foram considerados como 24 coeficientes, ao invés de 18.

Fazendo uma comparação com os valores obtidos por auto.arima e pelo processamento com AdaGrad (com *features* derivadas de auto.arima), pode-se observar que a nova estratégia (AdaGrad com *features* de ACF) é capaz de apresentar melhores resultados que os obtidos com anteriormente com esse algoritmo de DSM, em especial para horizontes mais distantes.

Em experimentos realizados com 90 séries do *dataset* M4, foi possível observar um ganho nos resultados de predição de cerca de 6,2% ao experimentar janelas deslizantes de 288 registros ao invés de 144 registros.

A ampliação da janela deslizante para 480 registros, no entanto, não resultou em melhora significativa.

Mais detalhes sobre esse experimento podem ser obtidos no *link* https://bit.ly/3oKoKwR (documento externo).

Fase 3 - EXPERIMENTO 3.3

Neste experimento, a técnica de *feature engineering* baseada em coeficientes de ACF foi avaliada para um conjunto de 48.000 séries de M4 e os valores obtidos por AdaGrad utilizado isoladamente foi comparada com a performance obtida com abordagens exploradas anteriormente (em que os atributos da série foram criados com o uso de auto.arima).

Calculando-se o ganho para o horizonte de 1 a 18 meses, obteve-se ganho = 23,31% com a técnica de ACF em comparação com os valores obtidos por AdaGrad anteriormente, sugerindo que a abordagem de *feature engineering* baseada no uso de coeficientes de autocorrelação é promissora.

Foi possível observar, também, que os valores obtidos por ADAGRAD_ACF isoladamente são mais próximos dos valores de ARIMA usado isoladamente do que os obtidos anteriormente por AdaGrad.

Mais detalhes sobre esse experimento podem ser obtidos no *link* https://bit.ly/2HLiFQe (documento externo).

Fase 3 - EXPERIMENTO 3.4

Nessa fase, foram efetuados testes com séries de diferentes frequências de M4 (mensais, diárias, horárias, trimestrais, semanais e anuais). Para isso, *features* da série que apresentavam valores referentes a ano e mês da observação (aplicáveis para as séries mensais avaliadas até então), foram substituídas por um número de ordem sequencial capaz de ser aplicado para todas as séries, independente do conhecimento ou não de informações sobre a data de ocorrência de cada observação das séries.

Em relação a algoritmos de DSM nova análise foi efetuada realizando experimentos com AdaGrad, RandomRules e ARF-Reg. O algoritmo AdaGrad foi mantido por continuar apresentando melhores resultados (e ser mais rápido) que os demais algoritmos no cenário analisado.

Em relação aos estudos com coeficientes de ACF, foram efetuados testes com janelas deslizantes de 72, 144, 288, 300, 360 e 480 observações e número de coeficientes ACF iguais a 12, 18, 24, 36 e 48. Os testes foram executados com 10 séries de cada tipo. Testes adicionais com 100 e 1000 séries foram efetuados com séries mensais.

Os menores valores de sMAPE obtidos por AdaGrad ocorrem em 50% dos casos com o uso de 12 coeficientes ACF (nCoefACF=12). Vale observar que o melhor desempenho para nCoefACF=12 foi observado para as séries mensais tanto nos testes com 10 séries, quanto nos testes com 100 e 1000 séries. No entanto, como os testes foram efetuados com apenas 10 séries de cada frequência, é possível que os valores observados não sejam

representativos.

Ou seja, deve-se avaliar a ampliação do estudo para um conjunto maior de séries a fim de poder oficializar os parâmetros a serem utilizados no processamento dos diferentes *datasets*. Essa oficialização deve ocorrer tanto em relação aos parâmetros de AdaGrad quanto em relação aos parâmetros de auto.arima.

Mais detalhes sobre esse experimento podem ser obtidos no *link* https://bit.ly/3mHfmrX (documento externo).

Fase 3 - EXPERIMENTO 3.5

Nesse experimento foram efetuados testes com 100 séries de cada uma das diferentes frequências de M4 (mensais, diárias, horárias, trimestrais, semanais e anuais), utilizando janelas deslizantes de 72, 144 e 288 observações e número de coeficientes ACF iguais a 2, 4, 6, 12, 18 e 24.

Pelos experimentos realizados, os menores valores de sMAPE para séries de frequências Diária, Horária, Mensal, Trimestral e Semanal foram obtidos com o uso de uma janela deslizante de 288 registros. Em relação ao número de coeficientes, os valores mais comumente observados para os melhores resultados foram 18 e 24 coeficientes ACF.

Para as séries Anuais, os resultados não foram conclusivos visto que, apesar de o universo de séries anuais ser de 23.000 séries, com comprimento variável entre 13 e 835 observações, para os experimentos foram selecionadas apenas séries anuais com 15 a 61 observações, impedindo verificar o comportamento dos resultados para tamanhos de janelas deslizantes superiores a 72 observações.

Mais detalhes sobre esse experimento podem ser obtidos no *link* https://bit.ly/2TDxLtk (documento externo).

Fase 3 - EXPERIMENTO 3.6

Nesta fase os experimentos foram estendidos para 200 séries de cada frequência de M4 (mensais, diárias, horárias, trimestrais, semanais e anuais), utilizadas janelas deslizantes de 288 registros e número de coeficientes de ACF iguais a 12, 18, 24, 36, 48 e 60.

Em relação aos valores obtidos com o processamento de auto.arima, os experimentos realizados nesta Fase 3 foram feitos considerando o uso de parâmetros *default* (*auto.arima(x)*) e parâmetros utilizados por *M4metalearning*, (*auto.arima(x, stepwise=FALSE, approximation= FALSE)*). Foi observado que o uso de parâmetros usados por *M4metalearning* não asseguram menor valor de sMAPE de teste de auto.arima para todos os tipos de série. Considerando que o tempo de processamento com parâmetros de auto.arima diferentes do *default* aumentam muito, os experimentos foram suficientes para determinar que auto.arima

deve ser usado no método proposto por este estudo usando sua configuração padrão. Vale destacar, para reforçar essa definição, que o tempo de processamento de auto.arima para 200 séries horárias de M4 com parâmetros *M4metalearning*, levou 52h17min (média de 15,68min/série) contra 5h33min (média de 1,67min/série) utilizado para a predição com auto.arima *default*. Além disso, nos testes com 200 séries anuais, o sMAPE de teste obtido com auto.arima padrão foi de 13,78% contra 16,40% obtido com auto.arima e parâmetros *M4metalearning*.

Sobre os experimentos com os coeficientes ACF, no processamento de séries semanais foi identificado que o uso de 48 coeficientes ACF foi o que apresentou melhor resultado para esse tipo de série. Isso sugere que pode ser relevante para essas séries avaliar o conjunto de coeficientes que representem cerca de 1 ano de histórico de observações.

De uma forma geral, em relação aos experimentos com os diferentes tipos de séries, não foi possível encontrar um número de coeficientes padronizado para todos os tipos de séries. Os números que ocorreram mais frequentemente foram 18 e 24 coeficientes, sendo o número de coeficientes iguais a 18 o que ocorreu em mais casos. Por esse motivo, os parâmetros padronizados para o método em relação ao processo de *feature engineering* baseado em coeficientes de correlação são os seguintes: **Janela deslizante = 288 observações; Número de coeficientes ACF = 18**.

Mais detalhes sobre esse experimento podem ser obtidos no *link* https://bit.ly/3jTNZt7 (documento externo).

Fase 3 - EXPERIMENTO 3.7

Os experimentos desta etapa foram realizados com as séries do *dataset* M3 com o intuito de fazer uma validação geral da abordagem de uso de janelas deslizantes de 288 observações e do uso de 18 coeficientes de autocorrelação.

Além disso, uma abordagem de fusão de *forecasts* foi experimentada e observado que, quando aplicada a séries selecionadas para AdaGrad pelo método, o resultado obtido pode ser melhor que o alcançado pelas técnicas avaliadas anteriormente. Nessa abordagem, ao selecionar uma série com base no erro de treinamento, se AdaGrad apresentar o menor erro nessa etapa, então sugere-se que os valores finais gerados pelo método sejam calculados com base numa média dos *forecasts* de auto.arima e AdaGrad.

Com o uso dessa técnica de **fusão** foram possíveis alcançar os seguintes ganhos positivos pelo método no processamento de séries do *dataset* M3 em relação aos valores obtidos com auto.arima usado isoladamente: 3,875% para séries mensais; 0,380% para séries trimestrais; 2,078% para séries do tipo Outras. Os resultados sugerem que é uma abordagem que deve ser considerada no método proposto pelo estudo.

Outra consideração importante sobre o método desenvolvido por este estudo, é que o

uso do valor de sMAPE de treino/Validação em nenhum cenário avaliado nos experimentos conseguiu melhorar o processo de seleção de séries de AdaGrad. Por isso, bem como pelo custo adicional de processamento que acarreta, o **processo de seleção adotado para o método levará em conta apenas os cálculos de erros de treinamento** realizados para as séries, sem necessidade de separação de dados para validação.

Mais detalhes sobre esse experimento podem ser obtidos no *link* https://bit.ly/3oMh0u8 (documento externo).