Big Data

Apache Spark - Pandas-on-Spark

Prof. Jean Paul Barddal

Agenda

Agenda

Pandas-on-Spark

- Recentemente, Spark recebeu uma interface baseada em Pandas para manipulação de dados
- Pandas é uma biblioteca muito famosa para tratamento de dados em Python e o uso de uma interface comum facilita o trabalho de Cientistas e Engenheiros de Dados

Pandas

- Biblioteca em Python para manipular dados tabulares
- Baseado em Numpy
- Diminui nossos esforços para extrair e manipular dados tabulares
- Visualização facilitada com matplotlib (e outras bibliotecas)
- Estruturas básicas: Series e DataFrame

Series

- Estrutura uni-dimensional com dados homogêneos
- Internamente, uma série é um array do numpy
- Uma Series tem características relevantes: nome, tipo de dados, índice e dados

Atenção

- Todo o material a seguir está focado em Pandas-on-Spark
- Muitos comandos são similares aos fornecidos no pandas e alguns idênticos
- Tome cuidado pois alguns comportamentos do Pandas podem não ser idênticos aos observados no Spark

Comandos básicos

- Um DataFrame possui atributos e funções bastante úteis
  - info
  - dtypes
  - len
  - shape
  - sample
  - columns
  - head
  - tail
  - drop
  - isna
  - sort_values

info

- Função info retorna as principais características de um DataFrame

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Tipos de Dados

- Para identificar os tipos de dados, podemos acessar o atributo **dtypes** do dataframe
- Exemplo:

```
df.dtypes
# PassengerId      int64
# Survived         int64
# Pclass           int64
# Name            object
# Sex             object
# Age            float64
# SibSp            int64
# Parch            int64
# Ticket          object
# Fare           float64
# Cabin           object
# Embarked        object
```

shape

- Atributo que retorna uma tupla contendo número de linhas e de colunas em um DataFrame

```
df.shape
# (891, 12)
linhas, colunas = df.shape
print(linhas) # 891
print(colunas) # 12
```

len

- Outra forma de obter a quantidade de linhas de um DataFrame

```
1 len(df)
2 # 891
```

sample

- Função que retorna uma amostra do DataFrame
- Duas formas principais de uso:
  - Definindo `n`: quantidade de elementos a serem retornados
  - Definindo `frac`: Fração do DataFrame a ser retornado

Definindo `n`:

```
df.sample(n=3)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
825	826	0	3	Flynn, Mr. John	male	NaN	0	0	368323	6.950	NaN	Q
537	538	1	1	LeRoy, Miss. Bertha	female	30.0	0	0	PC 17761	106.425	NaN	C
666	667	0	2	Butler, Mr. Reginald Fenton	male	25.0	0	0	234686	13.000	NaN	S

Definindo `frac`:

```
df.sample(frac=0.01)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
746	747	0	3	Abbott, Mr. Rossmore Edward	male	16.0	1	1	C.A. 2673	20.2500	NaN	S
604	605	1	1	Homer, Mr. Harry ("Mr E Haven")	male	35.0	0	0	111426	26.5500	NaN	C
482	483	0	3	Rouse, Mr. Richard Henry	male	50.0	0	0	A/5 3594	8.0500	NaN	S
303	304	1	2	Keane, Miss. Nora A	female	NaN	0	0	226593	12.3500	E101	Q
856	857	1	1	Wick, Mrs. George Dennick (Mary Hitchcock)	female	45.0	1	1	36928	164.8667	NaN	S
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5000	NaN	S
462	463	0	1	Gee, Mr. Arthur H	male	47.0	0	0	111320	38.5000	E63	S
574	575	0	3	Rush, Mr. Alfred George John	male	16.0	0	0	A/4. 20589	8.0500	NaN	S
86	87	0	3	Ford, Mr. William Neal	male	16.0	1	3	W./C. 6608	34.3750	NaN	S

columns

- Atributo que armazena o nome das colunas de um DataFrame

```
df.columns.values
# array(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex',
    'Age', 'SibSp',
#        'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
    dtype=object)
```

head

- Função que retorna as `n` **primeiras** linhas de um DataFrame

```
df.head(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

tail

- Função que retorna as `n` **últimas** linhas de um DataFrame

```
df.tail(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

Removendo linhas

- O comando **drop** permite remover linhas de acordo com o índice
- Para alterar a tabela original, use `inplace=True`

```
df.drop(0, axis=0)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

890 rows × 12 columns

Removendo colunas

- O comando **drop** também permite remover colunas, sendo necessário especificar `axis=1`
- Novamente, para alterar a tabela original, use `inplace=True`

```
df.drop('PassengerId', axis=1)
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 11 columns

Identificando valores faltantes

- Assim como em Series, os comandos **isna**/**isnull** permitem identificar valores faltantes
- O resultado destes comandos é uma máscara booleana que determina se valores estão ausentes ou não
- Normalmente é combinado com o comando **sum**, que permite calcular a quantidade de valores faltantes por coluna

Identificando valores faltantes

```
df.isna()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
5	False	False	False	False	False	True	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

890 rows × 12 columns

```
df.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Selecionando colunas

- A partir de um DataFrame, podemos selecionar uma ou mais colunas específicas para trabalharmos
- Ao selecionar apenas uma coluna, recebemos uma Series
- Selecionando duas ou mais colunas, temos um novo DataFrame
  - **Cuidado**: para selecionar duas ou mais colunas, precisamos especificar uma lista com os nomes

Selecionando colunas

```
df['PassengerId']
```

```
0        1
1        2
2        3
3        4
4        5
      ...
886    887
887    888
888    889
889    890
890    891
Name: PassengerId, Length: 891, dtype: int64
```

```
df[['Name', 'Survived']]
```

	Name	Survived
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1
2	Heikkinen, Miss. Laina	1
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1
4	Allen, Mr. William Henry	0
5	Moran, Mr. James	0
...	...	...
886	Montvila, Rev. Juozas	0
887	Graham, Miss. Margaret Edith	1
888	Johnston, Miss. Catherine Helen "Carrie"	0
889	Behr, Mr. Karl Howell	1
890	Dooley, Mr. Patrick	0

890 rows × 2 columns

Ordenação

- O comando **sort_values** permite realizar a ordenação de um DataFrame de acordo com uma ou mais colunas

```
df.sort_values('Name', ascending=True)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
845	846	0	3	Abbing, Mr. Anthony	male	42.0	0	0	C.A. 5547	7.5500	NaN	S
746	747	0	3	Abbott, Mr. Rossmore Edward	male	16.0	1	1	C.A. 2673	20.2500	NaN	S
279	280	1	3	Abbott, Mrs. Stanton (Rosa Hunt)	female	35.0	1	1	C.A. 2673	20.2500	NaN	S
308	309	0	2	Abelson, Mr. Samuel	male	30.0	1	0	P/PP 3381	24.0000	NaN	C
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	NaN	C
...	...	...	...	...	...	...	...	...	...	...	...	...
286	287	1	3	de Mulder, Mr. Theodore	male	30.0	0	0	345774	9.5000	NaN	S
282	283	0	3	de Pelsmaeker, Mr. Alfons	male	16.0	0	0	345778	9.5000	NaN	S
361	362	0	2	del Carlo, Mr. Sebastiano	male	29.0	1	0	SC/PARIS 2167	27.7208	NaN	C
153	154	0	3	van Billiard, Mr. Austin Blyler	male	40.5	0	2	A/5. 851	14.5000	NaN	S
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5000	NaN	S

891 rows × 12 columns

```
df.sort_values(['Name', 'Embarked'], ascending=[True, True])
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
845	846	0	3	Abbing, Mr. Anthony	male	42.0	0	0	C.A. 5547	7.5500	NaN	S
746	747	0	3	Abbott, Mr. Rossmore Edward	male	16.0	1	1	C.A. 2673	20.2500	NaN	S
279	280	1	3	Abbott, Mrs. Stanton (Rosa Hunt)	female	35.0	1	1	C.A. 2673	20.2500	NaN	S
308	309	0	2	Abelson, Mr. Samuel	male	30.0	1	0	P/PP 3381	24.0000	NaN	C
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	NaN	C
...	...	...	...	...	...	...	...	...	...	...	...	...
286	287	1	3	de Mulder, Mr. Theodore	male	30.0	0	0	345774	9.5000	NaN	S
282	283	0	3	de Pelsmaeker, Mr. Alfons	male	16.0	0	0	345778	9.5000	NaN	S
361	362	0	2	del Carlo, Mr. Sebastiano	male	29.0	1	0	SC/PARIS 2167	27.7208	NaN	C
153	154	0	3	van Billiard, Mr. Austin Blyler	male	40.5	0	2	A/5. 851	14.5000	NaN	S
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5000	NaN	S

891 rows × 12 columns

Filtragem de Dados

- Assim como em Series, podemos realizar filtragem (seleção) de dados de acordo com condições

```
mascara = (df['Age'] > 60) & (df['Sex'] == 'male')
df[mascara]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
33	34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.5000	NaN	S
54	55	0	1	Ostby, Mr. Engelhart Cornelius	male	65.0	0	1	113509	61.9792	B30	C
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
116	117	0	3	Connors, Mr. Patrick	male	70.5	0	0	370369	7.7500	NaN	Q
170	171	0	1	Van der hoef, Mr. Wyckoff	male	61.0	0	0	111240	33.5000	B19	S
252	253	0	1	Stead, Mr. William Thomas	male	62.0	0	0	113514	26.5500	C87	S
280	281	0	3	Duane, Mr. Frank	male	65.0	0	0	336439	7.7500	NaN	Q
326	327	0	3	Nysveen, Mr. Johan Hansen	male	61.0	0	0	345364	6.2375	NaN	S
438	439	0	1	Fortune, Mr. Mark	male	64.0	1	4	19950	263.0000	C23 C25 C27	S
456	457	0	1	Millet, Mr. Francis Davis	male	65.0	0	0	13509	26.5500	E38	S

Exercício

- A base de dados **Iris** contempla informações sobre três tipos de flores da família Iris: Setosa, Versicolor e Virginica
- A base de dados está disponível em: `https://jpbarddal.github.io/assets/data/datascience/iris.csv`
- Dada a base de dados, responda:
  1. Quantas instâncias (linhas) temos de cada tipo de flor? (R.: 50)
  2. Quantas flores possuem comprimento de sépala maior que 5? (R.: 118)
  3. Quantas flores do tipo Virginica possuem largura de sépala (sepal width) maior que 3? (R.: 17)
  4. Quantas flores existem do tipo Setosa ou Virginica com comprimento de pétala maior ou igual a 2? (R.: 0)