Ciência de Dados

UNIVERSIDADE CANDIDO MENDES | EAD

Introdução ao Big Data

- Utilização de dados com Pandas
- Visualização de dados com Matplotlib e Seaborn
  - pip install pandas
  - pip install numpy
  - pip install matplotlib
  - pip install seaborn

  - import pandas as pd
  - import numpy as np
  - from matplotlib import pyplot as plt
  - Import seaborn as sns

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Criando um dataframe a partir de um arquivo csv:
    - https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html
    - Exemplo
    - df = pd.read_csv(<filepath>, *sep* = ',', *parse_dates* = ['date'])

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10934 entries, 0 to 10933
Data columns (total 22 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   epi_week                                10934 non-null  int64
 1   date                                    10934 non-null  object
 2   country                                 10934 non-null  object
 3   state                                   10934 non-null  object
 4   city                                    10934 non-null  object
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10934 entries, 0 to 10933
Data columns (total 22 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   epi_week                                10934 non-null  int64
 1   date                                    10934 non-null  datetime64[ns]
 2   country                                 10934 non-null  object
 3   state                                   10934 non-null  object
 4   city                                    10934 non-null  object
```

- Alguns atributos importantes
- filepath: endereço do arquivo a ser carregado
- sep : str, default ' , ' (vírgula)
- usecols : lista de nomes/número das colunas que serão utilizadas
- parse_dates: lista de nomes/números das colunas que serão identificadas com data
- decimal: str, default '.' (ponto)
- encoding: str, optional (ex.: 'utf8')
- header: int, list of int, default 'infer', número da linha que contém os dados de cabeçalho
- index_col: int, str, sequence of int / str, or False, default, indica a coluna que deverá ser considerada como indexador das linhas no dataframe

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Efetuar uma cópia do dataframe:
      - df.copy()
    - Imprimir as primeiras linhas do dataframe:
      - df.head() // df.head(10)
    - Imprimir as últimas linhas do dataframe:
      - df.tail() // df.tail(10)
    - Imprimir uma amostra aleatória do dataframe:
      - df.sample() // df.sample(10)
    - Imprimir informações do dataframe:
      - df.info()
    - Imprimir informações estatísticas do dataframe:
      - df.describe()
    - Imprimir contagem de valores do dataframe:
      - df.count()
      - Caso existam valores Null, estes não serão contados
    - Imprimir contagem de valores do dataframe:
      - df.shape
      - Retorna uma tupla com o tamanho da matriz do dataframe

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Modificando o dataframe, eliminando colunas desnecessárias e linhas com valores nulos:
    - Substituindo valores nulos por 0:
      - df_raw_subset['bedrooms'] = df_raw_subset['bedrooms'].replace(np.nan, 0)
      - df_raw_subset['garage'] = df_raw_subset['garage'].replace(np.nan, 0)
      - df_raw_subset['tax'] = df_raw_subset['tax'].replace(np.nan, 0)
    - É possível também substituir pela média, basta trocar o número 0 por:
      - df_raw_subset['bedrooms'].mean()
    - Removendo valores nulos:
      - df_raw_subset = df_raw_subset[df_raw_subset['sales_price'].notna()]
      - df_raw_subset = df_raw_subset[df_raw_subset['area'].notna()]
    - Removendo colunas desnecessárias:
      - df_raw_subset.drop(['zip_code'], *axis*=1, *inplace*=True)
    - Removendo linhas com erros:
      - df_raw_subset.drop(*index* = df_raw_subset.query('area <= 9.00').index, *inplace*=True)
      - df_raw_subset.drop(*index* = df_raw_subset.nsmallest(5, 'sales_price', *keep*='all').index, *inplace*=True)

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Tipos de dados no Pandas:

Pandas dtype	Python type	NumPy type	Usage
object	str or mixed	string_, unicode_, mixed types	Text or mixed numeric and non-numeric values
int64	int	int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64	Integer numbers
float64	float	float_, float16, float32, float64	Floating point numbers
bool	bool	bool_	True/False values
datetime64	datetime	datetime64[ns]	Date and time values
timedelta[ns]	NA	NA	Differences between two datetimes
category	NA	NA	Finite list of text values

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Converter objetos para datetime:
      - df['date'] = pd.to_datetime(df['date'], *errors* = 'raise', *format*='%Y-%m-%d')
    - Converter para números:
      - df['newDeaths'] = df['newDeaths'].astype('float')
      - df['newDeaths'] = pd.to_numeric(df['newDeaths'], *errors* = 'raise', *downcast* = 'integer')
    - Na criação do dataframe: dtype

```
columns_names = ['date', 'country', 'state', 'city', 'newDeaths', 'newCases']
tipos_dados = {'newDeaths': 'float64', 'newCases': 'float64'}
df = pd.read_csv(r'https://raw.githubusercontent.com/wcota/covid19br/master/cases-brazil-states.csv', sep  = ',', usecols = columns_names, dtype = tipos_dados, parse_dates = ['date'])
```

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
  - Converter para números com a aplicação de uma função:

	sales_price	area	bedrooms	bathrooms	garage	tax	zip_code
0	R$ 610.000	50m²	1	1	0.00	R$ 390	22250210
1	R$ 1.020.000	74m²	2	2	1.00	None	22260040
2	R$ 1.735.000	93m²	3	1	1.00	R$ 1.371	22280110
3	R$ 800.000	88m²	3	1	0.00	R$ 440	22260140
4	R$ 229.000	23m²	None	1	0.00	R$ 540	22250040

```
def clean_prices(value):
    price_pattern = r'([\d,.]+)'
    if value is None:
        return None
    else:
        value_extracted = re.findall(price_pattern, value)[0]
        value_floated = value_extracted.replace('.','').replace(',','.')
        return value_floated
```

```
df_raw_subset['sales_price'] = df_raw_subset['sales_price'].apply(lambda x: clean_prices(x))
df_raw_subset['tax']   = df_raw_subset['tax'].apply(lambda x: clean_prices(x))
```

```
df_raw_subset['area'] = df_raw_subset['area'].apply(lambda x: x if x is None else x.replace('m²', ''))
```

```
df_raw_subset['bedrooms'] = df_raw_subset['bedrooms'].str.extract(r'(\d+)')
```

```
df_raw_subset['bathrooms'] = df_raw_subset['bathrooms'].str.extract(r'(\d+)')
```

- Existem colunas nesse dataset que não não é possível somente utilizando os métodos do Pandas, então criam-se funções utilizando expressões regulares para corrigir os dados:

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Visualizar os valores únicos de uma coluna:
      - `df['state'].unique()`
    - Contar a quantidade de linhas através dos valores únicos de uma coluna:
      - `pd.value_counts(df['state’])`
    - Localizar dados a partir de valores utilizando método .loc():
      - `df.loc[(df['state'] == 'RJ') & (df['date'] >= datetime.datetime(2021,4,1))]`
      - `df_raw_subset.loc[df_raw_subset['bedrooms']==5]`
      - `df_raw_subset.loc[(df_raw_subset['bedrooms']==5) & (df_raw_subset['bathrooms']==5)]`
      - `df_raw_subset.loc[(df_raw_subset['bedrooms']==3) | (df_raw_subset['bedrooms']==4)]`
      - `E => & // OU => |`

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Visualizar os valores únicos de uma coluna:
      - df['state'].unique()
    - Contar a quantidade de linhas através dos valores únicos de uma coluna:
      - pd.value_counts(df['state'])
    - Localizar dados a partir de valores utilizando método .loc():
      - df.loc[(df['state'] == 'RJ') & (df['date'] >= datetime.datetime(2021,4,1))]
      - df_raw_subset.loc[df_raw_subset['bedrooms']==5]
      - df_raw_subset.loc[(df_raw_subset['bedrooms']==5) & (df_raw_subset['bathrooms']==5)]
      - df_raw_subset.loc[(df_raw_subset['bedrooms']==3) | (df_raw_subset['bedrooms']==4)]
      - E => & // OU => |
    - Localizar dados a partir de valores utilizando método .query():
      - df_raw_subset.query('sales_price > 1000000')
      - df.query('state == "RJ" & date >= "2021-4-1"')
      - day_minus_six = date.today() - timedelta(days = 6)
      - df.query('state == "RJ" & date >= @day_minus_six')
    - Localizar dados a partir de valores utilizando o método .eval():
      - df_raw_subset[pd.eval('df_raw_subset["sales_price"] > 1000000')]

https://medium.com/horadecodar/data-science-tips-02-como-usar-loc-e-iloc-no-pandas-fab58e214d87

https://medium.com/horadecodar/como-usar-o-query-do-pandas-fdf4a00727dc

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Ordenando valores utilizando o método sort_values():
      - df_raw_subset.sort_values(*by*='sales_price', *ascending*=False).head(15)
      - df_raw_subset.sort_values(*by*=['sales_price', 'area'], *ascending*=False).head(15)
    - 15 dias com mais mortes pela COVID-19 no Brasil:
      - df.nlargest(15, 'newDeaths')
    - 15 dias com mais mortes pela COVID-19 no Brasil?
      - df.query('state == "TOTAL"').sort_values(*by*=['newDeaths'], *ascending*=False).head(15)
    - 15 dias com mais mortes pela COVID-19 em SP?
      - df.query('state == "SP"').sort_values(*by*=['newDeaths'], *ascending*=False).head(15)
      - df.query('state == "SP"').nlargest(15, 'newDeaths')

Introdução ao Big Data

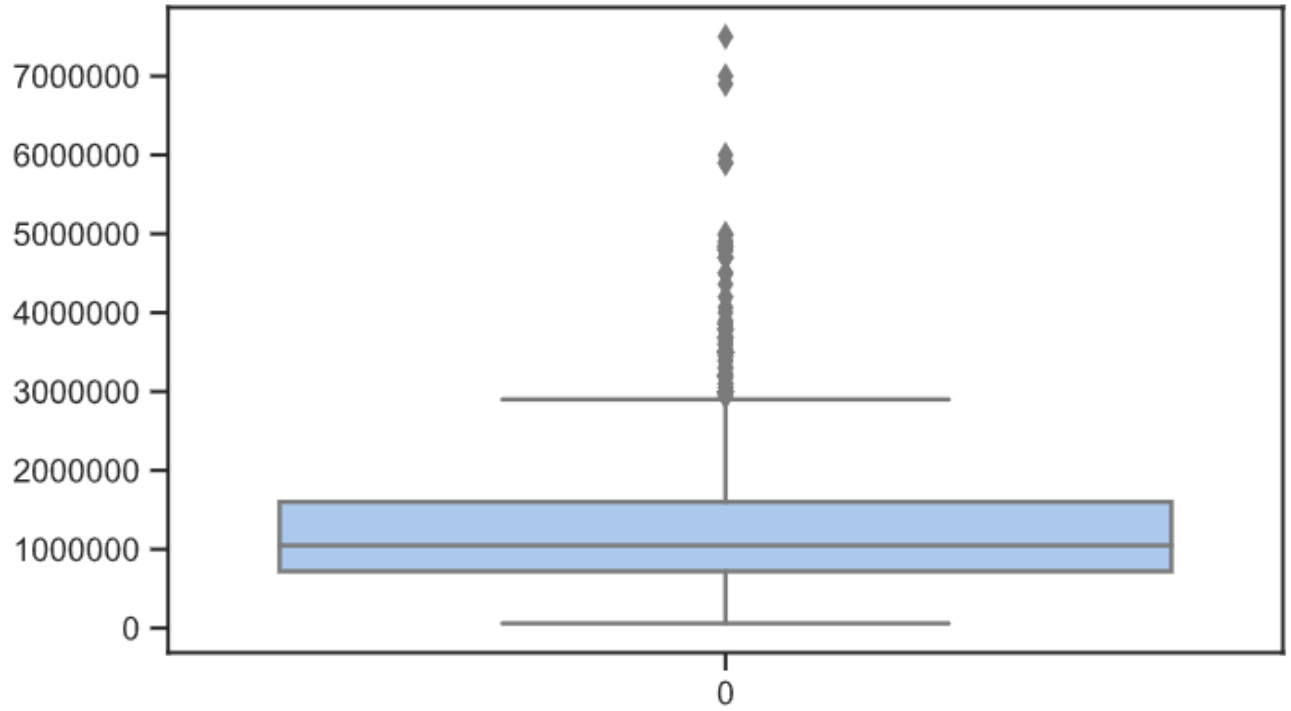
- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Como encontrar outliers?
      - Utilizar o cálculo de IRQ:

```
q1_preco = round((df_raw_subset.sales_price.quantile(q = 0.25)),2)
q3_preco = round((df_raw_subset.sales_price.quantile(q = 0.75)),2)
print('1º quartil:', q1_preco)
print('3º quartil:', q3_preco)
irq = round((q3_preco - q1_preco), 2)
print('IRQ:', irq)
iqr_li = round((q1_preco - (1.5 * irq)), 2)
iqr_ls = round((q3_preco + (1.5 * irq)), 2)
print('Limite inferior IRQ:', iqr_li)
print('Limite superior IRQ:', iqr_ls)
```

      - Filtrar valores acima dos limites:
        - df_raw_subset.loc[df_raw_subset['sales_price'] >= iqr_ls].sort_values(*by*=['sales_price'], *ascending* = False)

Introdução ao Big Data

- Análise de dados com Python + Pandas:
  - Lista de comandos:
    - Como encontrar outliers?
      - Utilizar o gráfico de boxplot:
      - `sns.boxplot(data = df_raw_subset['sales_price'])`



UNIVERSIDADE
CANDIDO
MENDES

EAD