Roteiro de Exploratory Data Analysis - EDA

Este roteiro contempla os principais elementos utilizados em uma análise de dados, para exploração, entendimento, organização e definições sobre o conjunto de dados.

**O conjunto de dados**

Será utilizado o conjunto Automobile Sales Data, coletado no Kaggle. O arquivo encontra no GitHub relativo ao curso, na pasta EDA. A seguir algumas informações importantes sobre as *features* que compõem o conjunto.

|  |  |
| --- | --- |
| **ORDERNUMBER** | This column represents the unique identification number assigned to each order. |
| **QUANTITYORDERED** | It indicates the number of items ordered in each order. |
| **PRICEEACH** | This column specifies the price of each item in the order. |
| **ORDERLINENUMBER** | It represents the line number of each item within an order. |
| **SALES** | This column denotes the total sales amount for each order, which is calculated by multiplying the quantity ordered by the price of each item. |
| **ORDERDATE** | It denotes the date on which the order was placed. |
| **DAYS\_SINCE\_LASTORDER** | This column represents the number of days that have passed since the last order for each customer. It can be used to analyze customer purchasing patterns. |
| **STATUS** | It indicates the status of the order, such as "Shipped," "In Process," "Cancelled," "Disputed," "On Hold," or "Resolved." |
| **PRODUCTLINE** | This column specifies the product line categories to which each item belongs. |
| **MSRP** | It stands for Manufacturer's Suggested Retail Price and represents the suggested selling price for each item. |
| **PRODUCTCODE** | This column represents the unique code assigned to each product. |
| **CUSTOMERNAME** | It denotes the name of the customer who placed the order. |
| **PHONE** | This column contains the contact phone number for the customer. |
| **ADDRESSLINE1** | It represents the first line of the customer's address. |
| **CITY** | This column specifies the city where the customer is located. |
| **POSTALCODE** | It denotes the postal code or ZIP code associated with the customer's address. |
| **COUNTRY** | This column indicates the country where the customer is located. |
| **CONTACTLASTNAME** | It represents the last name of the contact person associated with the customer. |
| **CONTACTFIRSTNAME** | This column denotes the first name of the contact person associated with the customer. |
| **DEALSIZE** | It indicates the size of the deal or order, which are the categories "Small," "Medium," or "Large." |

Para a prática, vamos utilizar esse roteiro como base para consulta aos comandos. Os comandos devem ser escritos e não copiados, para evitar problemas com a fonte, caracteres, etc, quando colados.

O instrutor irá guia-los com os comandos, explicando e mostrando os exemplos e avaliações que podem ser feitas.

import pandas as pd

df = pd.read\_csv('Auto Sales data.csv') **ou** df = pd.read\_csv(‘caminho’)  
Considerações: (..., parse\_dates=['column'], dayfirst=True)

**Manipulação básica de Dataframe**

df.head()  
pd.unique()  
df.info()  
df['column'].head()  
df[['column1', 'column2']].head()

df.loc[row]  
df.loc[rowM:rowN]  
df.loc[rowM:rowN, ['column1', 'column2']]

**Definição de uma coluna como a “principal”**

df.set\_index('column').head()  
df.set\_index('column', inplace=True)   
**Obs:** inplace é útil quando um índice 0,1,2,etc refere-se a um elemento apenas de uma coluna.  
df.loc['item'] #*item* dentro da coluna que virou *index*df['column'].head()

**Criação de uma lista ou um numpy a partir de um Dataframe**

df['column'].to\_list()  
df['column'].to\_numpy()

**...visualização**

column = df['column'].to\_numpy()  
N = len(column)  
plt.scatter(range(N), column)

plt.xlabel('xlabel')

plt.ylabel('ylabel')

plt.title('title');

df.sort\_values('column', inplace=True) #lembrar que *inplace* mantém no df original

df.drop('column', axis=1)  
df.drop('index', axis=0)

**Ordenação dos dados**

df.sort\_values('column', inplace=True) #útil para visualizar relacionamento condicional

**Criar uma nova coluna**

df['new\_column'] = some  
exemplo: from datetime import datetime  
 temp\_date = datetime(2020, 6, 1)  
 df.['DAYS\_SINCE\_LASTORDER'] = (temp\_date – df['ORDERDATE']).dt.days

**As vezes não precisamos de dados de um tipo específicos para a análise. Categóricos, numéricos que não são informativos, etc.**

df.select\_dtypes(inlcude=['Dtype', 'Dtype'])  
Drop nos selecionados  
df\_num = df.select\_dtypes(inlcude=['Dtype', 'Dtype']).drop(columns=['column'])  
Ver as colunas restantes, daqueles tipos especificados  
print(df\_num.columns, len(list(df\_num.columns.values)))

**Descrição de um Dataframe**

df.describe() #para numéricos (padrão)  
df.select\_dtypes(include=['object']).describe() #para objetos  
Para visualização “transposta”, usar .describe().T

**Valores faltantes e duplicados**

df.isnull()  
df.isnull().sum()  
df.duplicated() #considera todas as colunas, pra identificar se as linhas são iguais  
df.duplicated(subset=['column']) #uma coluna específica ou várias colunas

**Consultas condicionais**

df.query('column CONDITION value')  
df.query('column CONDITION @variable')  
CONDITION: <, <=, >, >=, ==, !=

**Visualizações**

df.plot.bar()  
df['column'].value\_counts().plot.bar()  
df['column'].plot.hist()  
df['column'].plot.pie()  
exemplo: df.loc[1:5, ['column']].plot.pie(subplots=True)

**Visualizações com Plotly**

fig = px.bar('dataframe', x='index', y='values')  
exemplo: count = df['STATUS'].value\_counts()  
 fig = px.bar(count, x=value\_counts.index, y=value\_counts.values)  
 fig.show()  
fig = px.pie(values='values')  
exemplo numérico:   
 percentage = df.loc[1:5, ['SALES']] / df.loc[1:5, ['SALES']].sum() \* 100  
 perc = percentage['SALES'].to\_numpy()  
 fig = px.pie(values=perc, hole=0.3)  
 fig.show()  
exemplo categórico:  
 percentage = df.loc[1:5, ['PRODUCTLINE']].value\_counts() / df.loc[1:5, ['PRODUCTLINE']].value\_counts().sum() \* 100  
 fig = px.pie(values=percentage, hole=0.3)  
 fig.show()

fig = go.Figure(go.Funnel(y=index, x=values))  
fig.show()  
exemplo: con\_count = df['COUNTRY'].value\_counts()[:5]  
 fig = go.Figure(go.Funnel(y=con\_count.index, x=con\_count.values))  
 fig.update\_layout(template='plotly\_white')  
 fig.show()

**Visualizações com Seaborn**

sns.histplot(df['column'], bins=numbins) #kde=True para *kernel density estimation*plt.show()  
sns.boxplot(df['column'])  
plt.show()  
sns.pairplot(data=numeric\_data)  
exemplo: df2 = df.select\_dtypes(['int64', 'float64']).drop(columns=['ORDERNUMBER'])  
 sns.pairplot(data=df2, corner=True)

sns.lineplot(data=dataframe, x='column', y='column')  
observações: hue=agrupamento para ter linhas de cores diferentes (exemplo 'Year'). A separação pode ser por dia, mês, ano, etc.  
exemplo: df['Year']=df['ORDERDATE'].dt.year  
 sns.lineplot(data=df, x='ORDERDATE', y='SALES', hue='Year')  
 plt.show()

sns.barplot(data=dataframe, x='column', y='column')  
observações: hue=agrupamento para ter barras de cores diferentes (exemplo 'DEALSIZE').  
exemplo: sns.barplot(data=df, x='PRODUCTLINE', y='SALES', hue='DEALSIZE')  
 plt.show()

**Visualização com MatplotLib**

plt.fill\_between(dataframe.x, dataframe.y)  
observações: pode separar os valores por where= (por exemplo, cruzamento dos eixos ou valores das curvas) e definir cores diferentes com o color=.  
exemplo: df['SELLING\_PRICE\_DIFF'] = df['PRICEEACH']-df['MSRP']  
 plt.fill\_between(df['ORDERDATE'], df['SELLING\_PRICE\_DIFF'], color='red', where=( df['SELLING\_PRICE\_DIFF'] < 0))  
 plt.fill\_between(df['ORDERDATE'], df['SELLING\_PRICE\_DIFF'], color='green', where=( df['SELLING\_PRICE\_DIFF'] >= 0))  
 plt.show()

**Para analisar!**Ordene a coluna *ORDERDATE* e reexecute o Fill\_Between. O que acontece com o resultado visual?

**Cumulative Distribution Function (CDF)**

cdf = np.cumsum(dataseries) / np.sum(dataseries)  
plt.plot(dataseries, cdf)  
exemplo: x = df.sort\_values('SALES')['SALES'].to\_numpy()  
 cdf\_x = np.cumsum(x)/np.sum(x)  
 plt.plot(x, cdf\_x)

É interessante plotar histograma e CDF no mesmo gráfico.

exemplo: fig, ax = plt.subplots()  
 ax.plot(x, cdf\_x, color='red')  
 ax2 = ax1.twinx()  
 ax2.hist(x, bins=20, alpha=0.7) #tente trocar a ordem do plot e hist!

**BoxPlot**

plt.boxplot(data)  
exemplo: a = df['MSRP']  
 plt.boxplot(a)  
 ax = df['MSRP'].plot.kde() #Para avaliar conjuntamente  
 plt.hist(a) #Para avaliar conjuntamente

**ATIVIDADE**

Um exemplo de criação de tabela, a partir do conjunto completo:

df['PRODUCTCODE'].unique()  
len(df['PRODUCTCODE'].unique()  
product\_df = df[['PRODUCTCODE', 'PRODUCTLINE', 'MSRP']]  
product\_df.drop\_duplicates(inplace=True)  
product\_df.reset\_index(drop=True, inplace=True)

Esta sequência irá criar um novo *dataframe* com apenas 3 colunas. As linhas, serão mantidas apenas se não forem duplicadas. Compare a quantidade de linhas do novo *dataframe* (.*shape*) após o *drop\_duplicate* com a quantidade de valores *unique*.

Agora, faça o mesmo para as seguintes colunas: ORDERNUMBER, PRODUCTCODE, CUSTOMERNAME, ORDERDATE, ORDERLINENUMBER, QUANTITYORDERED, PRICEEACH, SALES, DAYS\_SINCE\_LASTORDER, DEALSIZE, STATUS, criando um novo *dataframe* para colunas relacionadas a *orders*.