Análise Exploratória de Dados

cognitiza

Your Data & AI school

Prof. Altino Dantas

Análise Exploratória de Dados

Em estatística, análise exploratória de dados é uma metodologia de análise de dados para sintetizar as principais características deles, geralmente, usando gráficos estatísticos e outros métodos de visualização de dados.

Análise Exploratória de Dados

Abordagem/filosofia para análise de dados que emprega uma variedade de técnicas (principalmente gráficas) para:

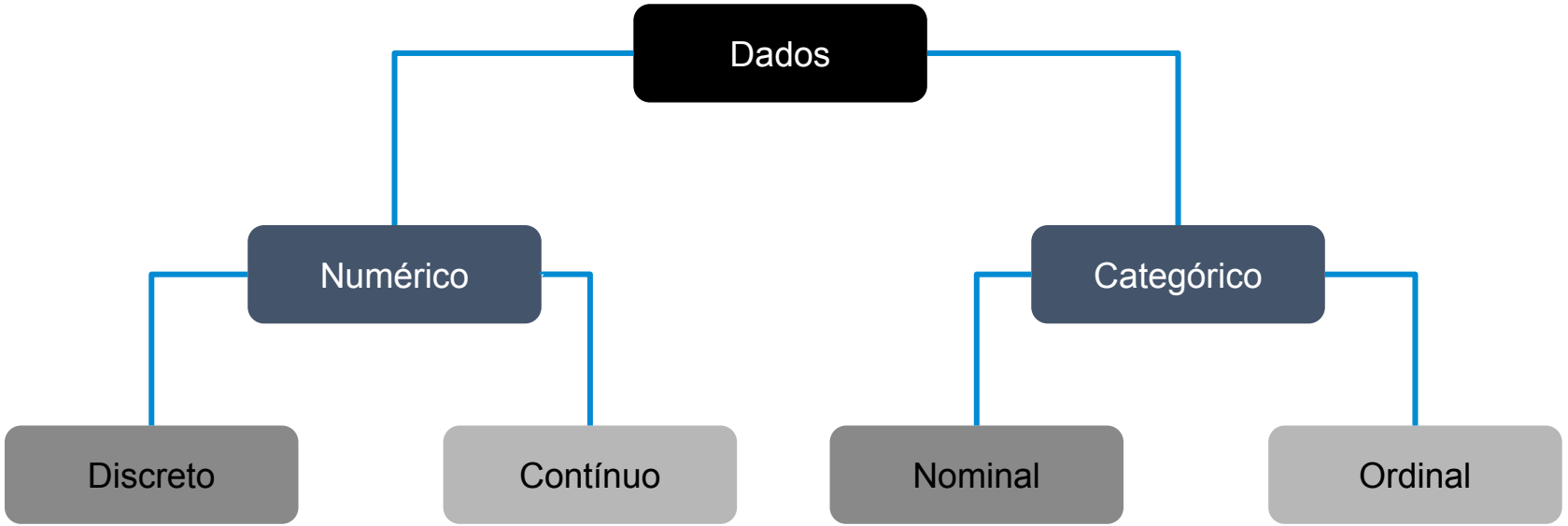
- Maximizar o entendimento sobre um conjunto de dados;
- Revelar informações escondidas nos dados;
- Extrair variáveis importantes;
- Detectar *outliers* e anomalias;
- Testar suposições;
- Desenvolver modelos de inferência adequados.

Análise Exploratória de Dados

As técnicas gráficas empregadas em EDA são, frequentemente, bastante simples:

- Construir gráficos com os **dados brutos**
  - Gráficos de barras, histogramas, gráficos de probabilidade, gráficos de dispersão, gráfico de caixa etc.
- Obter **estatística descritiva**
  - Gráficos de média, gráficos de desvio padrão, gráficos de caixa.
- Posicionar esses gráficos de modo a maximizar nossas habilidades naturais de **reconhecimento de padrões**
  - Uso de vários gráficos por página;
  - Combinar diferentes fontes de dados em um mesmo gráfico.

Tipos de dados



EDA na prática com Python

- Dataset obtido no repositório de Machine Learning da University of California, Irvine (UCI);
- Trata-se de dados sobre empresas fraudulentas;

Data Set Characteristics:	Multivariate	**Number of Instances:**	777	**Area:**	N/A
Attribute Characteristics:	Real	**Number of Attributes:**	18	**Date Donated**	2018-07-14
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	53618

https://archive.ics.uci.edu/ml/datasets/Audit+Data

EDA na prática com Python

- Importar bibliotecas para comodidade:

```
1 # bibiliotecas de manipulação de dados
2 import pandas as pd
3 # bibliotecas de visualização de dados
4 import seaborn as sns
5 import matplotlib.pyplot as plt
```

EDA na prática com Python

- Obtendo e verificando o formato dos dados

```
1 # carregar o arquivo dos dados
2 df = pd.read_csv("trial.csv", sep=",")
```

```
1 # verificar o formando do dataset
2 df.shape
```

```
(776, 18)
```

Temos **776 instâncias** carregadas, além do cabeçalho dos nossos dados, o qual contém os nomes dos atributos. Além de **18 atributos** em nosso conjunto de dados, bem como foi informado no repositório da UCI.

EDA na prática com Python

- Visualizando os dados

```
df.head()
```

	Sector_score	LOCATION_ID	PARA_A	SCORE_A	PARA_B	SCORE_B	TOTAL	numbers	Marks	Money_Value	MONEY_Marks	District	Loss	LOSS_SCORE	History	History_score	Score	Risk
0	3.89	23	4.18	6	2.50	2	6.68	5.0	2	3.38	2	2	0	2	0	2	2.4	1
1	3.89	6	0.00	2	4.83	2	4.83	5.0	2	0.94	2	2	0	2	0	2	2.0	0
2	3.89	6	0.51	2	0.23	2	0.74	5.0	2	0.00	2	2	0	2	0	2	2.0	0
3	3.89	6	0.00	2	10.80	6	10.80	6.0	6	11.75	6	2	0	2	0	2	4.4	1
4	3.89	6	0.00	2	0.08	2	0.08	5.0	2	0.00	2	2	0	2	0	2	2.0	0

Podemos perceber que os dados são todos numéricos, o que facilitará em um posterior processo de modelagem dos dados. Para garantir, podemos utilizar o comando ***info()***

EDA na prática com Python

- Visualizando os dados

```
# visualizar atributos e seus tipos
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 776 entries, 0 to 775
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Sector_score   776 non-null    float64
 1   LOCATION_ID    776 non-null    object
 2   PARA_A         776 non-null    float64
 3   SCORE_A        776 non-null    int64
 4   PARA_B         776 non-null    float64
 5   SCORE_B        776 non-null    int64
 6   TOTAL          776 non-null    float64
 7   numbers        776 non-null    float64
 8   Marks          776 non-null    int64
 9   Money_Value    775 non-null    float64
 10  MONEY_Marks    776 non-null    int64
 11  District       776 non-null    int64
 12  Loss           776 non-null    int64
 13  LOSS_SCORE     776 non-null    int64
 14  History        776 non-null    int64
 15  History_score  776 non-null    int64
 16  Score          776 non-null    float64
 17  Risk           776 non-null    int64
dtypes: float64(7), int64(10), object(1)
memory usage: 109.2+ KB
```

Assim, podemos perceber que existem **7 atributos do tipo float64**, ou seja, valores decimais; **10 inteiros do tipo int64**; e um do tipo **object**.

O atributo **LOCATION_ID**, apesar de parecer que continha apenas valores inteiros, apresenta outros valores em sua estrutura. Vamos analisá-lo mais cuidadosamente.

EDA na prática com Python

- Investigando o tipo de dado

```
# verificar a quantidade de ocorrências por valor existente
df['LOCATION_ID'].value_counts().sort_index()
```

Como os dados não estão obedecendo a ordem esperada para inteiros, estes dados foram tratados como *string;*

```
1          11
11         26
12         47
13         35
14         20
15         35
16         52
17          1
18         16
19         68
    ...
41          1
42          1
43          7
44          1
5          44
6          33
7           4
8          76
9          53
LOHARU      1
NUH         1
SAFIDON     1
Name: LOCATION_ID, dtype: int64
```

EDA na prática com Python

- Convertendo formato de dados

```
1 # transformar os dados em numéricos
2 df['LOCATION_ID'] = pd.to_numeric(df['LOCATION_ID'], errors='coerce')
3
4 # verificar atributos com valores ausentes
5 df.isnull().sum(axis = 0)
```

```
Sector_score      0
LOCATION_ID       3
PARA_A            0
SCORE_A           0
PARA_B            0
SCORE_B           0
TOTAL             0
numbers           0
Marks             0
Money_Value       1
MONEY_Marks       0
District          0
Loss              0
LOSS_SCORE        0
History           0
History_score     0
Score             0
Risk              0
dtype: int64
```

- "coerce" no parâmetro erros atribui "nan" para o registro sem valor.
- Depois de converter o tipo do atributo LOCATION_ID, checamos se existem valores nulos no dataset.
- Percebe-se que duas variáveis possuem valores nulos: LOCATION_ID e Money_Value

EDA na prática com Python

- Lidando com valores ausentes

```
# substituir valores ausentes de 'LOCATION_ID'
df['LOCATION_ID'] = df['LOCATION_ID'].fillna(-1)

# verificar a substituição
print(df["LOCATION_ID"].iloc[351])
print(df["LOCATION_ID"].iloc[355])
print(df["LOCATION_ID"].iloc[367])
```

```
-1.0
-1.0
-1.0
```

```
# substituir valores ausentes de 'Money_Value'
df['Money_Value'] = df['Money_Value'].where(pd.notna(df['Money_Value']), df['Money_Value'].mean())
```

```
# verificar a substituição
print(df['Money_Value'].iloc[642])
print(df['Money_Value'].iloc[642] == df['Money_Value'].mean())
```

```
14.13763096774195
True
```

EDA na prática com Python

- Agrupando, agregando, ordenando e cortando

```
1 df.groupby('LOCATION_ID')['Score'].mean().sort_values(ascending=False).head(10)
```

```
LOCATION_ID
41.0    4.400000
24.0    4.200000
34.0    4.200000
42.0    4.000000
1.0     3.927273
38.0    3.800000
40.0    3.733333
7.0     3.550000
20.0    3.360000
30.0    3.350000
Name: Score, dtype: float64
```


Operação de agregação

EDA na prática com Python

- Balanceamento

```
1 # verificar o balanceamento dos dados
2 df["Risk"].value_counts(normalize=True)
```

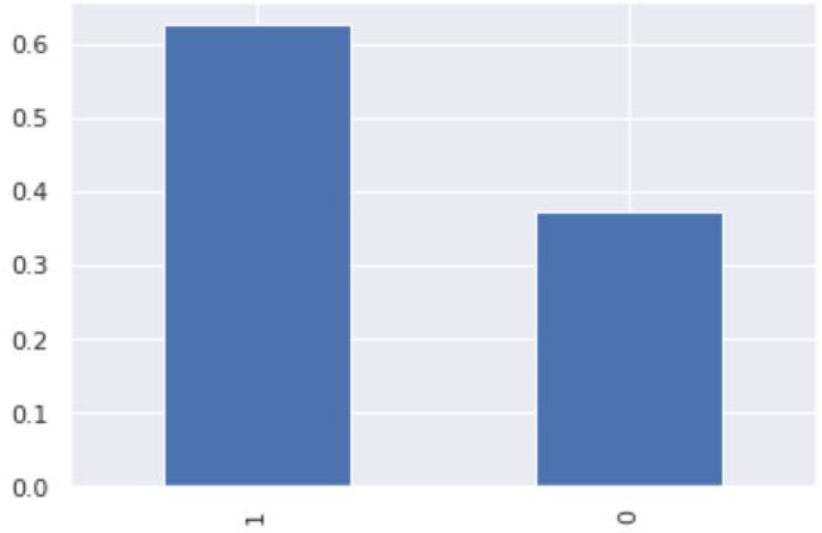
```
1    0.626289
0    0.373711
Name: Risk, dtype: float64
```

Percebemos que **62,6% das empresas** analisadas apresentariam alguma **irregularidade** em uma auditoria, enquanto que as demais, **37,4%, não**. Desta forma, sabemos que as amostras estão de certa forma balanceadas.

EDA na prática com Python

- Balanceamento

```
1 # plotando gráfico de barras
2 df["Risk"].value_counts(normalize=True).plot.bar();
```



Objetos de dados Pandas (DataFrame e Series) possuem o método **plot()** como meio de plot rápido para gráficos. Consulte a documentação completa aqui.

EDA na prática com Python

- Estatística descritiva

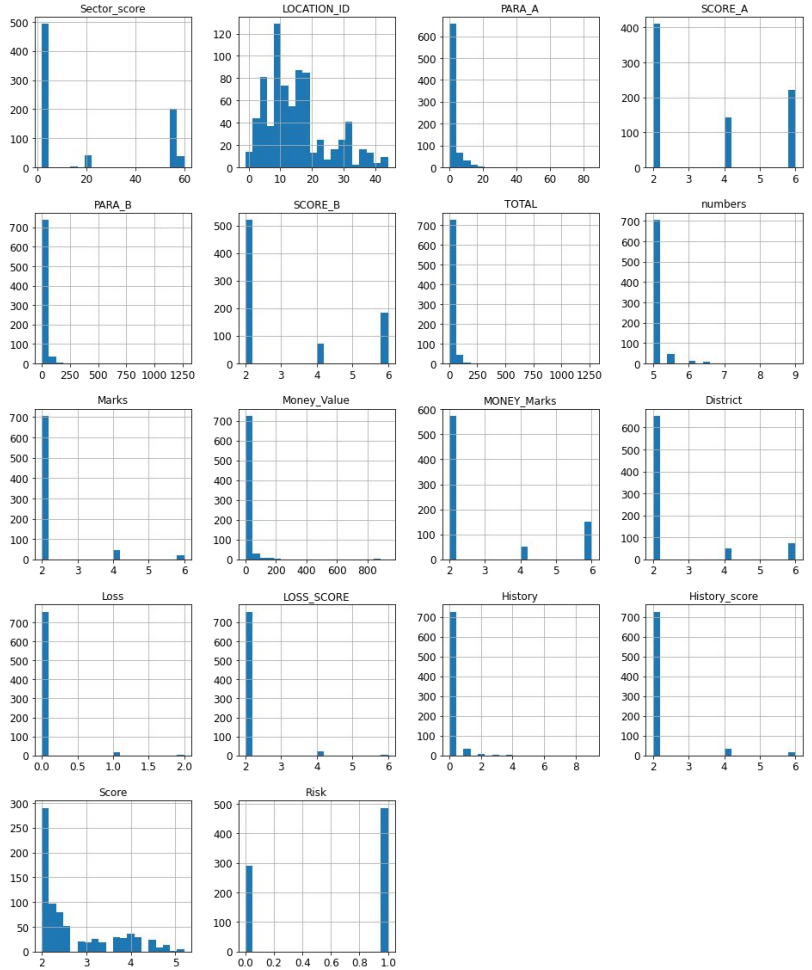
```
1 # visualizar estatísticas descritivas
2 df.describe()
```

	Sector_score	LOCATION_ID	PARA_A	SCORE_A	PARA_B	...	Loss	LOSS_SCORE	History	History_score	Score	Risk
count	776.000000	776.000000	776.000000	776.000000	776.000000		776.000000	776.000000	776.000000	776.000000	776.000000	776.000000
mean	20.184536	14.795103	2.450194	3.512887	10.799988		0.029639	2.061856	0.104381	2.167526	2.702577	0.626289
std	24.319017	9.921136	5.678870	1.740549	50.083624		0.184280	0.375080	0.531031	0.679869	0.858923	0.484100
min	1.850000	-1.000000	0.000000	2.000000	0.000000		0.000000	2.000000	0.000000	2.000000	2.000000	0.000000
25%	2.370000	8.000000	0.210000	2.000000	0.000000		0.000000	2.000000	0.000000	2.000000	2.000000	0.000000
50%	3.890000	13.000000	0.875000	2.000000	0.405000		0.000000	2.000000	0.000000	2.000000	2.400000	1.000000
75%	55.570000	19.000000	2.480000	6.000000	4.160000		0.000000	2.000000	0.000000	2.000000	3.250000	1.000000
max	59.850000	44.000000	85.000000	6.000000	1264.630000		2.000000	6.000000	9.000000	6.000000	5.200000	1.000000

EDA na prática com Python

- Visualizando a distribuição dos dados

```
# visualizar distribuição das variáveis
df.hist(figsize=(16, 20), bins=20, xlabelsize=12, ylabelsize=12);
```

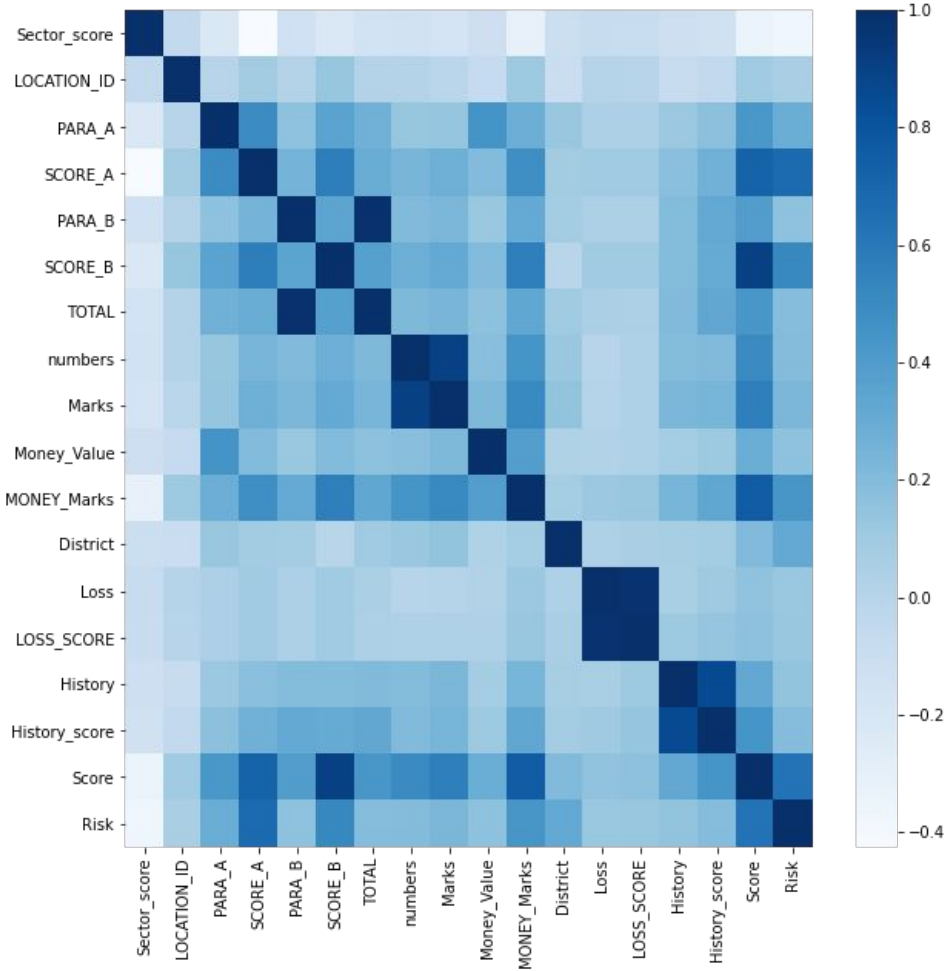


- Percebe-se que a maioria das variáveis possuem valores concentrados, com exceção de LOCATION_ID, Score, SCORE_A e SCORE_B.
- Outra correlação forte é entre Score e Risk.

EDA na prática com Python

- Visualizando a correlação

```
# visualizar correlação entre as variáveis
plt.figure(figsize=(10, 10))
sns.heatmap(df.corr(), cmap='Blues');
```



Quanto mais escura a célula, maior a correlação entre as variáveis. Por exemplo, PARA_B e TOTAL possuem uma forte correlação.

EDA na prática com Python

- Visualizando a correlação

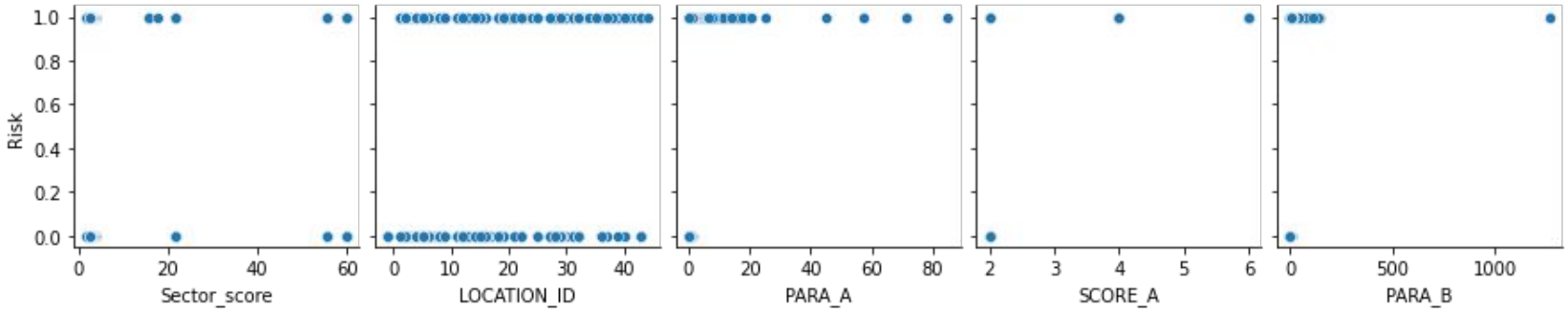
```
1 # verificar a correlação entre as variáveis e a variável alvo
2 df.corr()['Risk'][:-1].sort_values(ascending=False)
```

```
SCORE_A          0.671863
Score            0.632268
SCORE_B          0.515045
MONEY_Marks      0.440226
District         0.317795
PARA_A           0.292425
Marks            0.228098
numbers          0.197750
TOTAL            0.190793
History_score    0.190466
PARA_B           0.162807
Money_Value      0.160543
History          0.151937
LOSS_SCORE       0.127472
Loss             0.124322
LOCATION_ID      0.056306
Sector_score    -0.374588
Name: Risk, dtype: float64
```

EDA na prática com Python

- Visualizando a correlação entre a variável alvo e outras

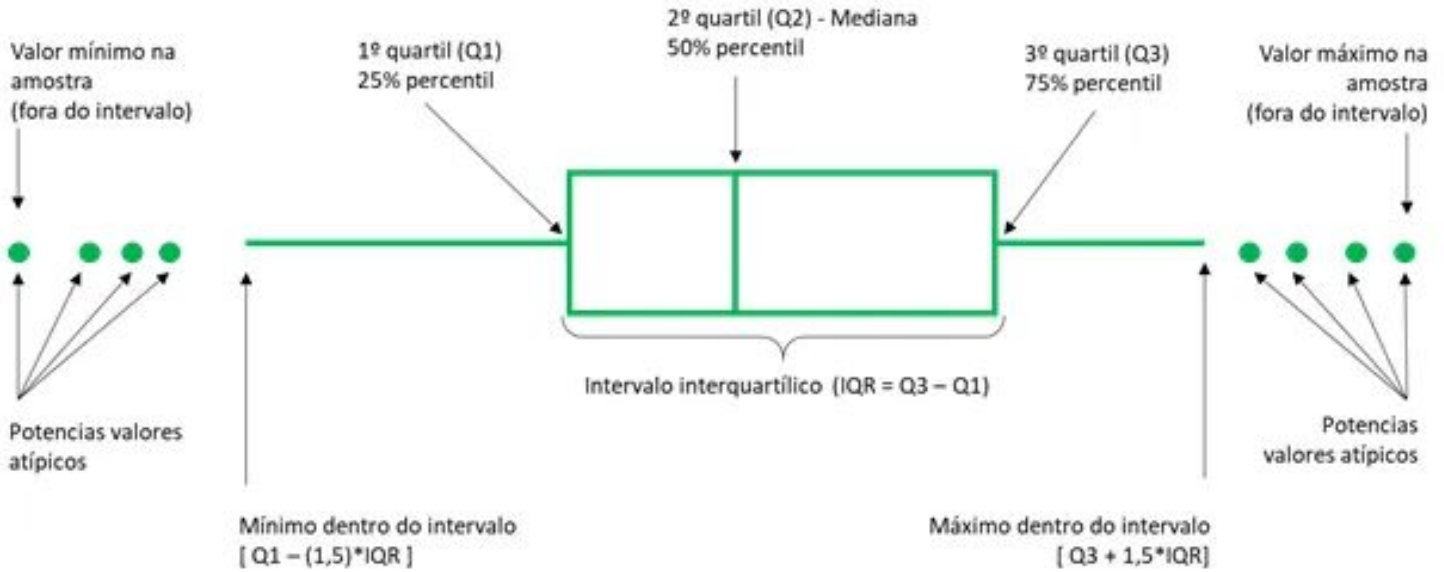
```
# visualizar a correlação entre as variáveis e a variável alvo
for i in range(0, len(df.columns), 5):
    sns.pairplot(data=df, x_vars=df.columns[i:i+5], y_vars=['Risk'])
```



Com exceção dos atributos Sector_score e LOCATION_ID é possível notar que há uma separação nos valores do atributo entre os valores da nossa classe (Risk). Para Risk com valor 0 não há ocorrências de instâncias com valor levemente superior a 2.

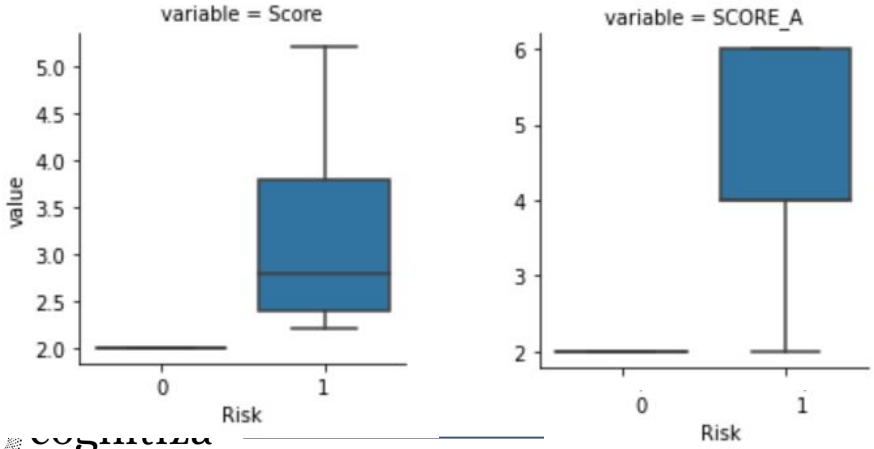
Entendendo Boxplot

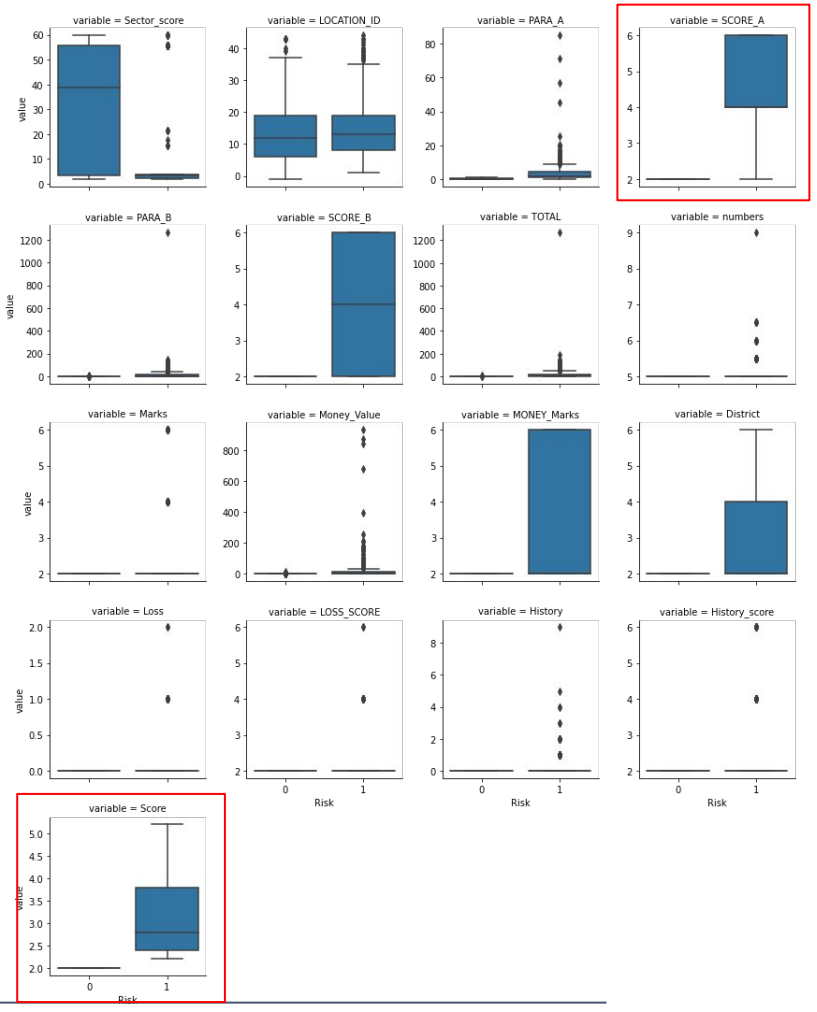
Boxplot é utilizado para avaliar e comparar o formato, **tendência central** e **variabilidade de distribuições** de amostra, e para **procurar outliers**. Por padrão, um boxplot demonstra a mediana, os quartis, o intervalo interquartil(IQR) e outliers para cada variável.



EDA na prática com Python

```
# visualizar a distribuição dos dados para cada variável
df_melt = df.melt(id_vars=["Risk"])
grid = sns.axisgrid.FacetGrid(df_melt[df_melt.variable.isin(df.columns)],
                              col='variable', col_wrap=4, sharey=False)
grid.map(sns.boxplot, "Risk","value", order=None);
```





Medidas específicas

```
# média, median, moda, variância e desvio padrão
df['SCORE_B'].mean()
df['SCORE_B'].median()
df['SCORE_B'].mode()
df['SCORE_B'].var()
df['SCORE_B'].std()

# quartis
df['SCORE_B'].quantile(0.25)
df['SCORE_B'].quantile(0.5)
df['SCORE_B'].quantile(0.75)
```

Exercício - Grupo

Realize uma análise exploratória dos dados do dataset “*dados-delivery.csv”* e responda as seguintes questões:

1. *Quais são os produtos mais pedidos?*
2. *Qual é o tempo médio de atraso para cada produto cuja entrega atrasa?*
3. *Qual a taxa média de atraso nas entregas (percentual de aumento em relação ao previsto)?*
4. *Quais produtos mais atrasam?*
5. *Quais dias da semana mais têm pedidos? Há correlação com atrasos?*
6. *Quais variáveis possuem correlação?*
7. *A maior parte dos pedidos são feitos em qual horário?*
8. *Quais campos possuem valores ausentes? É melhor preenchê-los ou descartá-los?*
9. *As avaliações dos pedidos podem ser explicadas pela variável atraso?*

Obrigado.