Técnicas de Pré-processamento

Prof(a). Giselly Alves Reis

Pré-processamento de dados

- Bases de dados;
- Tipos de dados;
- Exploração de dados;
- Eliminação manual;
- Integração de dados;
- Amostragem de dados;
- Dados desbalanceados;
- Limpeza de dados;
- Transformação de dados;
- Redução de dimensionalidade.

Bases de dados

- Dados abertos do Brasil (Saúde, Educação, Eleições, etc...)
  - https://basedosdados.org/
- UC Irvine Machine Learning Repository (Dados abertos para a comunidade de AM)
  - https://archive.ics.uci.edu/
- Dados abertos de países europeus, instituições, agências e organismos da UE
  - https://data.europa.eu/en

Tipos de Dados

- Quantitativo:
  - Contínuos: Podem assumir valores infinitos
    - Ex.: Valor da ação, Investimento, Custo
  - Discretos: Os valores são finitos e contáveis
    - Ex.: Idade, Peso, Altura
- Qualitativo: Expressam uma qualidade ou domínio
  - Ex.: {grande, pequeno, médio}, nome, sintoma

Tipos de Dados

Nome do Campo	Tipo de Dados	Quantitativo/ Qualitativo
Idade	Numérico	
Nome	Texto	
Nota de Satisfação	Numérico	
Data de Nascimento	Data	
Gênero	Texto	
Salário	Numérico	
Código Postal	Texto	
Classificação	Numérico	
Temperatura	Numérico	
Número de Filhos	Numérico	

Tipos de Dados

Nome do Campo	Tipo de Dados	Quantitativo/ Qualitativo
Idade	Numérico	Quantitativo
Nome	Texto	Qualitativo
Nota de Satisfação	Numérico	Quantitativo
Data de Nascimento	Data	Qualitativo
Gênero	Texto	Qualitativo
Salário	Numérico	Quantitativo
Código Postal	Texto	Qualitativo
Classificação	Numérico	Quantitativo
Temperatura	Numérico	Quantitativo
Número de Filhos	Numérico	Quantitativo

Escala dos dados

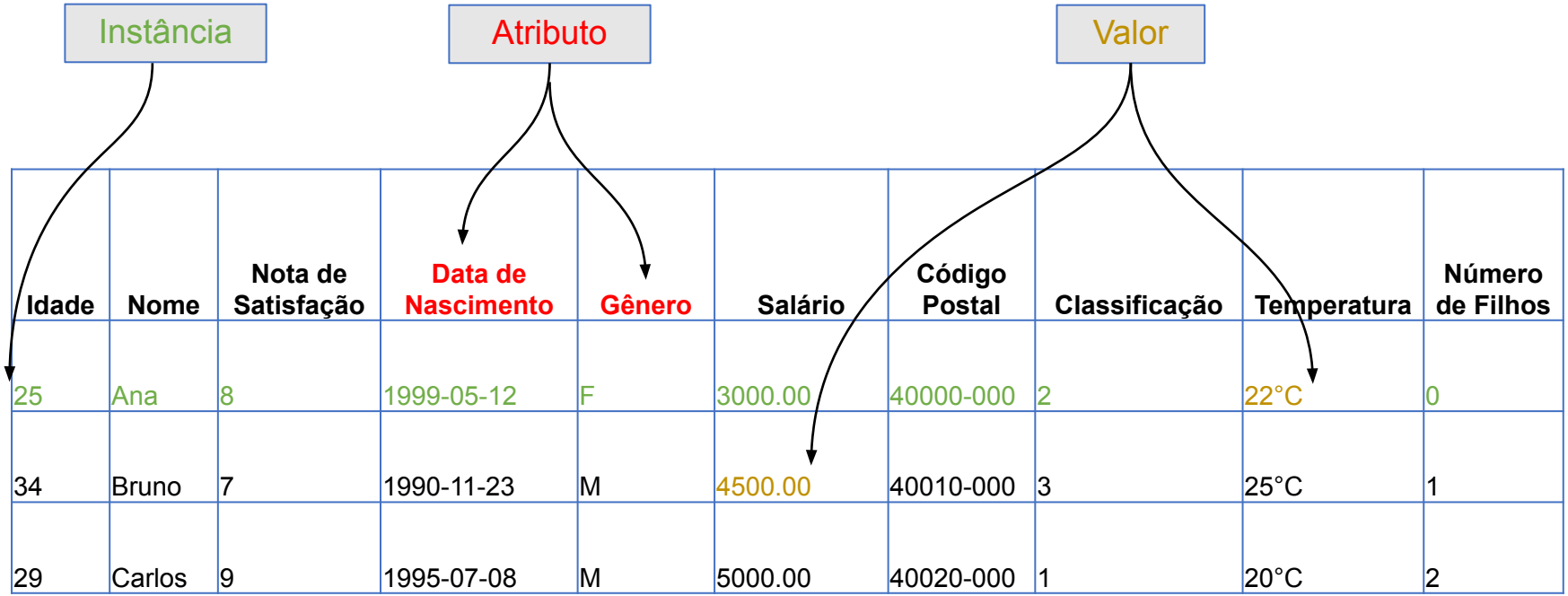
- Nominal:
  - Ex.: Nome, RG, CPF, CEP, Sexo.
- Ordinal:
  - Ex.: hierarquia, {frio, morno, quente}, {pequeno, médio, grande}
- Intervalar:
  - Ex.: Temperatura, Data de Nascimento
- Racional:
  - Ex.: Tamanho, distância, salário, saldo de conta

Escala dos dados

Nome do Campo	Tipo de Dados	Quantitativo/ Qualitativo	Escala dos dados
Idade	Numérico	Quantitativo	Racional
Nome	Texto	Qualitativo	Nominal
Nota de Satisfação	Numérico	Quantitativo	Ordinal
Data de Nascimento	Data	Qualitativo	Intervalar
Gênero	Texto	Qualitativo	Nominal
Salário	Numérico	Quantitativo	Racional
Código Postal	Texto	Qualitativo	Nominal
Classificação	Numérico	Quantitativo	Ordinal
Temperatura	Numérico	Quantitativo	Intervalar
Número de Filhos	Numérico	Quantitativo	Racional

Escala de Medição: Nominal – Ordinal – Intervalar - Racional

Técnicas de pré-processamento de dados



Idade	Nome	Nota de Satisfação	Data de Nascimento	Gênero	Salário	Código Postal	Classificação	Temperatura	Número de Filhos
25	Ana	8	1999-05-12	F	3000.00	40000-000	2	22°C	0
34	Bruno	7	1990-11-23	M	4500.00	40010-000	3	25°C	1
29	Carlos	9	1995-07-08	M	5000.00	40020-000	1	20°C	2

Objetivos da Exploração de dados

- Compreensão inicial:
  - Dimensões e Estrutura (analisar número de observações e variáveis)
  - Visualização Geral (gráficos e estatísticas)
- Identificação de tendências e padrões:
  - Correlações: (identificar possíveis dependências).
  - Distribuições: (identificar forma, centralidade e dispersão).
- Detecção de anomalias e valores ausentes:
  - Outliers: (desvios significativos do padrão).
  - Valores Faltantes: (extensão e impacto para imputação ou remoção).

Objetivos da Exploração de dados

- Seleção de características (feature selection):
  - Relevância: (para reduzir dimensionalidade/melhorar a eficiência).
- Preparação dos dados:
  - Normalização/Padronização: (melhorar o desempenho do modelo).
  - Transformações: (melhorar a distribuição dos dados).
- Hipóteses iniciais:
  - Formulação de Hipóteses: (baseado nas descobertas).

Benefícios da Exploração de dados

- Melhoria na qualidade dos modelos:
  - Seleção das melhores características e preparação mais adequada dos dados.
- Redução de custos e tempo:
  - Antecipação de problemas nos dados e otimização do processo de modelagem.
- Tomada de decisão informada:
  - As descobertas podem ajudar na escolha de algoritmos, parâmetros e estratégias de validação.

Exploração de dados

- Ex.: Série de preços das ações da IBM

```
[ ]  # Instalar a biblioteca yfinance se ainda não estiver instalada
     !pip install yfinance
```

```
[ ]  import yfinance as yf
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from scipy.stats import normaltest
```

Exploração de dados

```
[3]  # Baixar dados da IBM
     ibm = yf.Ticker("IBM")
     data = ibm.history(period="max")

     # Visualizar as primeiras linhas do dataset
     data.head()
```

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
1962-01-02 00:00:00-05:00	1.530518	1.530518	1.513321	1.513321	407940	0.0	0.0
1962-01-03 00:00:00-05:00	1.513321	1.526549	1.513321	1.526549	305955	0.0	0.0
1962-01-04 00:00:00-05:00	1.526549	1.526549	1.510675	1.511336	274575	0.0	0.0
1962-01-05 00:00:00-05:00	1.509352	1.509352	1.478927	1.481573	384405	0.0	0.0
1962-01-08 00:00:00-05:00	1.480250	1.480250	1.441888	1.453793	572685	0.0	0.0

Próximas etapas: Gerar código com data | Ver gráficos recomendados | New interactive sheet

Exploração de dados

Resumo Estatístico Básico

O método `describe()` fornece um resumo estatístico básico dos dados, incluindo média, desvio padrão, mínimo, máximo e quartis.

```
[4]  # Resumo estatístico básico
     data.describe()
```

	Open	High	Low	Close	Volume	Dividends	Stock Splits
count	15765.000000	15765.000000	15765.000000	15765.000000	1.576500e+04	15765.000000	15765.000000
mean	39.103166	39.457331	38.767292	39.117057	5.080366e+06	0.006631	0.000954
std	43.928192	44.276925	43.597611	43.946204	4.599632e+06	0.083014	0.046751
min	0.809812	0.850832	0.793933	0.809812	0.000000e+00	0.000000	0.000000
25%	4.099490	4.128819	4.075211	4.101435	1.551427e+06	0.000000	0.000000
50%	11.804594	11.904463	11.697225	11.801018	4.289646e+06	0.000000	0.000000
75%	65.113766	65.950524	64.282889	65.159279	7.018346e+06	0.000000	0.000000
max	193.941664	195.524994	193.714996	194.729996	7.263916e+07	1.670000	4.000000

Exploração de dados

Medidas de Tendência Central

As medidas de tendência central incluem a média, mediana e moda. Elas ajudam a entender onde os valores dos dados estão centralizados.

- **Média**: Valor médio dos dados.
- **Mediana**: Valor que separa a metade superior da metade inferior dos dados.
- **Moda**: Valor mais frequente nos dados.

```
[5]  # Medidas de tendência central
     print("Média:", data['Close'].mean())
     print("Mediana:", data['Close'].median())
     print("Moda:", data['Close'].mode()[0])
```

```
Média: 39.11705735370994
Mediana: 11.801017761230469
Moda: 3.289201498031616
```

Exploração de dados

Medidas de Dispersão

As medidas de dispersão incluem desvio padrão, variância e intervalo. Elas ajudam a entender o quão dispersos estão os valores dos dados.

- **Desvio Padrão**: Medida de dispersão dos dados em relação à média.
- **Variância**: Quadrado do desvio padrão.
- **Intervalo (máximo - mínimo)**: Diferença entre o valor máximo e mínimo.

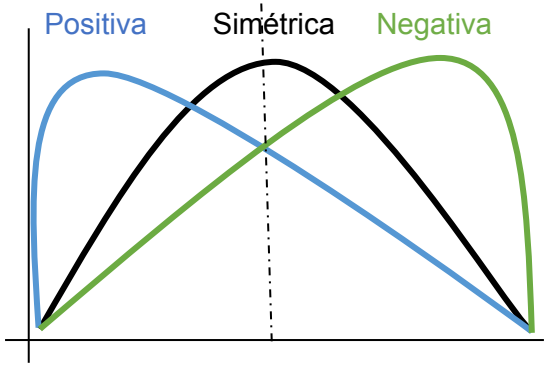
```
[6]  # Medidas de dispersão
     print("Desvio Padrão:", data['Close'].std())
     print("Variância:", data['Close'].var())
     print("Intervalo (máximo - mínimo):", data['Close'].max() - data['Close'].min())
```

```
Desvio Padrão: 43.94620409979413
Variância: 1931.2688547807622
Intervalo (máximo - mínimo): 193.9201835989952
```

Exploração de dados

- Medidas de Forma: Assimetria (Skewness)
  - Coeficiente de assimetria (b1): mede a falta de simetria de um conjunto de dados. Possui a fórmula:

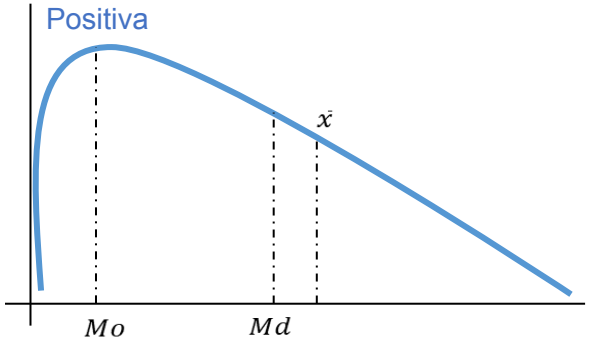
$$b1 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{X_i - \bar{X}}{S}\right)^3$$



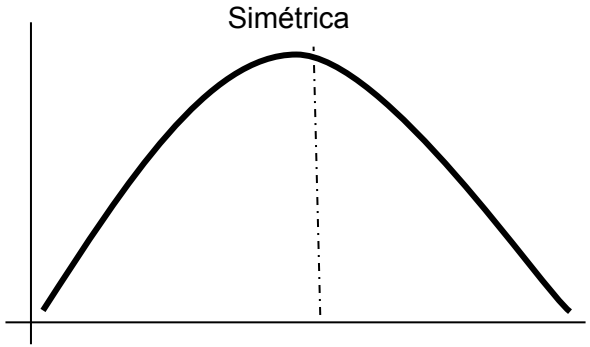
  - Assume valores:
    - b1 = 0; distribuição é simétrica;
    - b1 > 0; distribuição é assimétrica positiva ou à direita;
    - b1 < 0; distribuição é assimétrica negativa ou à esquerda.

Exploração de dados

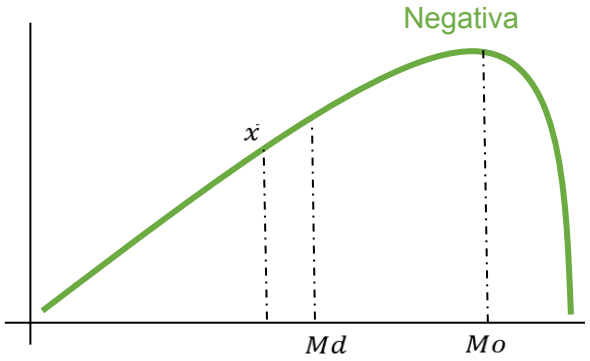
- Medidas de Forma: Assimetria (Skewness)



$Mo < Md < \bar{x}$



$Mo = Md = \bar{x}$



$Mo > Md > \bar{x}$

$Mo \rightarrow Moda$
$Md \rightarrow Mediana$
$\bar{x} \rightarrow Média$

Exploração de dados

- Medidas de Forma: Curtose
  - Medida que caracteriza o achatamento da curva dos dados em função da distribuição normal

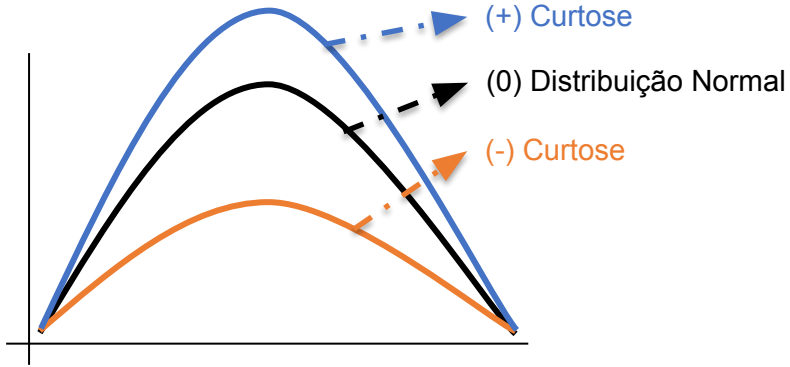
$$C = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{(n-1)\sigma^4} - 3$$

- C=0 (normal) indica que o histograma dos dados apresenta o mesmo achatamento que a distribuição normal;
- C < 0 (negativa) indica que o histograma dos dados apresenta uma distribuição mais achatada que a distribuição normal;
- C > 0 (positiva) indica que o histograma dos dados apresenta uma distribuição mais alta e concentrada que a distribuição normal.

Exploração de dados

- Medidas de Forma: Curtose
  - Medida que caracteriza o achatamento da curva dos dados em função da distribuição normal

$$C = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{(n-1)\sigma^4} - 3$$



  - C=0 (normal)
  - C < 0 (negativa)
  - C > 0 (positiva)

Exploração de dados

Medidas de Forma

As medidas de forma incluem assimetria (skewness) e curtose (kurtosis). Elas ajudam a entender a forma da distribuição dos dados.

- **Assimetria (Skewness)**: Mede o grau de assimetria da distribuição dos dados.
- **Curtose (Kurtosis)**: Mede o quão achatada ou alongada é a distribuição dos dados.

```
[7]  # Medidas de forma
     print("Assimetria (Skewness):", data['Close'].skew())
     print("Curtose (Kurtosis):", data['Close'].kurt())
```

```
Assimetria (Skewness): 1.0180857426619905
Curtose (Kurtosis): -0.15455271065420373
```

Exploração de dados

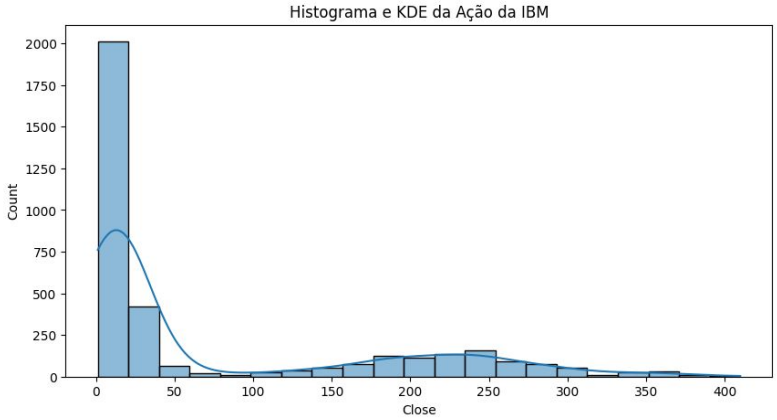
Gráficos

Histograma e KDE

O histograma e o KDE (Kernel Density Estimate) mostram a distribuição dos dados.

```
[9]  # Gráfico de normalidade (Histograma e KDE)
     plt.figure(figsize=(10, 5))
     sns.histplot(data['Close'], kde=True)
     plt.title('Histograma e KDE da Ação da IBM')
     plt.show()
```

1s

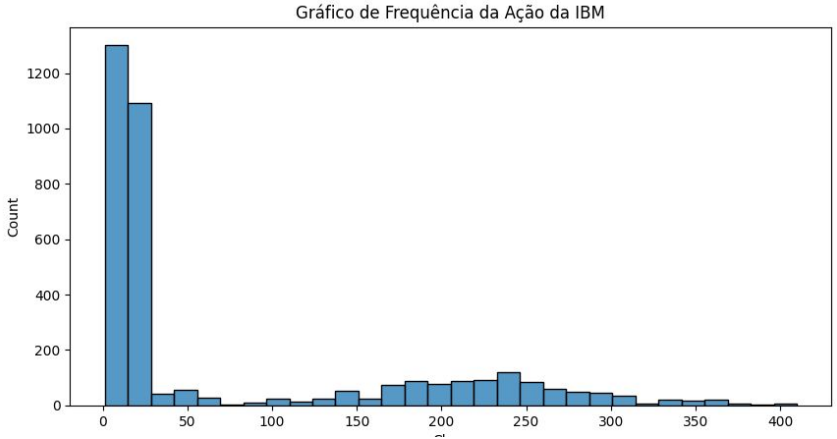


Exploração de dados

Gráfico de Frequência

O gráfico de frequência mostra a frequência dos valores.

```
# Gráfico de frequência
plt.figure(figsize=(10, 5))
sns.histplot(data['Close'], bins=30, kde=False)
plt.title('Gráfico de Frequência da Ação da IBM')
plt.show()
```



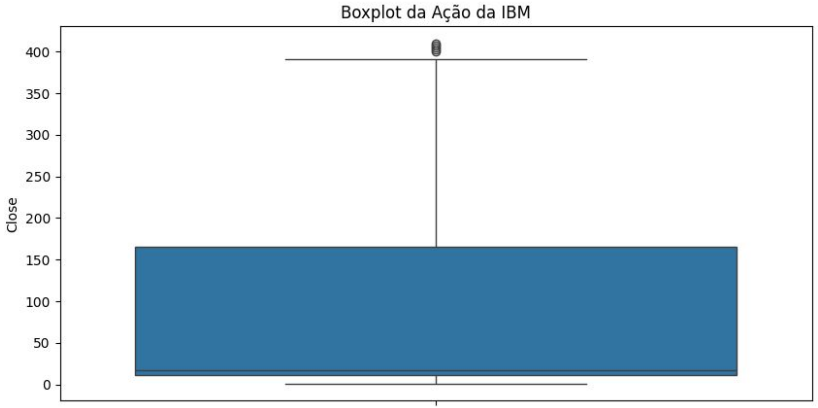
Exploração de dados

Boxplot

O boxplot mostra a mediana, quartis e possíveis outliers.

```
[12] # Boxplot
     plt.figure(figsize=(10, 5))
     sns.boxplot(x=data['Close'])
     plt.title('Boxplot da Ação da IBM')
     plt.show()
```

0s



Exploração de dados

Gráfico de Linha

O gráfico de linha mostra a tendência do preço de fechamento ao longo do tempo.

```
[13] # Gráfico de linha para visualizar a tendência ao longo do tempo
     plt.figure(figsize=(12, 6))
     plt.plot(data.index, data['Close'])
     plt.title('Preço de Fechamento ao Longo do Tempo')
     plt.xlabel('Data')
     plt.ylabel('Preço de Fechamento')
     plt.show()
```

Exercício

- Carregue o Dataset **load_breast_cancer**, da biblioteca **skit-learn** e faça a Análise exploratória dos dados.
- Converta os dados para um DataFrame do pandas
- Visualize as 10 primeiras linhas do Dataset;
- Faça um resumo das estatísticas básicas (describe);
- Escolha uma coluna, exemplo: “mean radius”, do Dataset e aplique os comandos abaixo:
  - Medidas de tendência central, dispersão (Desvio Padrão, Variância, Intervalo mín. x máx.);
  - Gere os gráficos de Histograma, Frequência, Boxplot e gráfico de Linha.

Arquivo: Aula_02_Analise_Exploratória_Load_Brest_Cancer_Exercício.ipynb

Eliminação manual

Comando	Descrição	Exemplo
dropna()	Remove linhas ou colunas com valores ausentes (NaN).	df.dropna()
drop()	Remove linhas ou colunas especificadas pelo índice ou nome.	df.drop(columns=['Coluna1'])
drop_duplicates()	Remove linhas duplicadas do DataFrame.	df.drop_duplicates()
filter()	Filtra linhas ou colunas com base em critérios específicos.	df.filter(items=['Coluna1', 'Coluna2'])
isin()	Filtra linhas onde os valores estão em uma lista especificada.	df[df['Coluna1'].isin([1, 2, 3])]

Eliminação manual

Comando	Descrição	Exemplo
query()	Filtra linhas usando uma expressão de consulta.	df.query('Coluna1 > 10')
loc[]	Acessa um grupo de linhas e colunas por rótulos.	df.loc[df['Coluna1'] > 10]
iloc[]	Acessa um grupo de linhas e colunas por posições inteiras.	df.iloc[0:5, 0:2]
where()	Substitui valores onde a condição é False.	df.where(df['Coluna1'] > 10)
mask()	Substitui valores onde a condição é True.	df.mask(df['Coluna1'] > 10)

Integração de dados

Comando	Descrição	Exemplo
merge()	Combina DataFrames com base em uma coluna comum.	pd.merge(df1, df2, on='ColunaComum')
concat()	Concatena DataFrames verticalmente ou horizontalmente.	pd.concat([df1, df2])
join()	Combina DataFrames com base em seus índices.	df1.join(df2, how='inner')
append()	Adiciona linhas de um DataFrame ao final de outro.	df1.append(df2)
combine_first()	Preenche valores ausentes em um DataFrame com valores de outro.	df1.combine_first(df2)

Integração de dados

Comando	Descrição	Exemplo
update()	Atualiza valores de um DataFrame com valores de outro.	df1.update(df2)
assign()	Adiciona novas colunas a um DataFrame.	df1.assign(NovaColuna=df1['Coluna1'] + df1['Coluna2'])
pivot()	Transforma linhas em colunas.	df.pivot(index='Coluna1', columns='Coluna2', values='Coluna3')
melt()	Transforma colunas em linhas.	df.melt(id_vars=['Coluna1'], value_vars=['Coluna2'])
groupby()	Agrupa dados com base em uma ou mais colunas.	df.groupby('Coluna1').sum()

Amostragem de dados

Comando	Descrição	Exemplo
sample()	Seleciona uma amostra aleatória de elementos do DataFrame.	df.sample(n=5)
head()	Retorna as primeiras n linhas do DataFrame.	df.head(10)
tail()	Retorna as últimas n linhas do DataFrame.	df.tail(10)
nsmallest()	Retorna as n linhas com os menores valores em uma coluna específica.	df.nsmallest(5, 'Coluna1')

Amostragem de dados

Comando	Descrição	Exemplo
nlargest()	Retorna as n linhas com os maiores valores em uma coluna específica.	df.nlargest(5, 'Coluna1')
query()	Filtra linhas usando uma expressão de consulta.	df.query('Coluna1 > 10')
sort_values()	Ordena o DataFrame por valores de colunas.	df.sort_values(by='Coluna1')
groupby()	Agrupa dados com base em uma ou mais colunas.	df.groupby('Coluna1').sample(n=2)

Limpeza de dados

Comando	Descrição	Exemplo
dropna()	Remove linhas ou colunas com valores ausentes (NaN).	df.dropna()
fillna()	Preenche valores ausentes com um valor específico ou método.	df.fillna(0)
drop_duplicates()	Remove linhas duplicadas do DataFrame.	df.drop_duplicates()
replace()	Substitui valores específicos em colunas.	df.replace({'Coluna1': {'ValorAntigo': 'ValorNovo'}})
astype()	Converte o tipo de dados de uma coluna.	df['Coluna1'] = df['Coluna1'].astype(int)

Limpeza de dados

Comando	Descrição	Exemplo
str.strip()	Remove espaços em branco e caracteres indesejados das strings.	df['Coluna1'] = df['Coluna1'].str.strip()
str.lower()	Converte strings para minúsculas.	df['Coluna1'] = df['Coluna1'].str.lower()
str.upper()	Converte strings para maiúsculas.	df['Coluna1'] = df['Coluna1'].str.upper()
str.replace()	Substitui substrings dentro de strings.	df['Coluna1'] = df['Coluna1'].str.replace('old', 'new')
apply()	Aplica uma função a uma coluna ou DataFrame.	df['Coluna1'] = df['Coluna1'].apply(lambda x: x + 1)

Dados desbalanceados

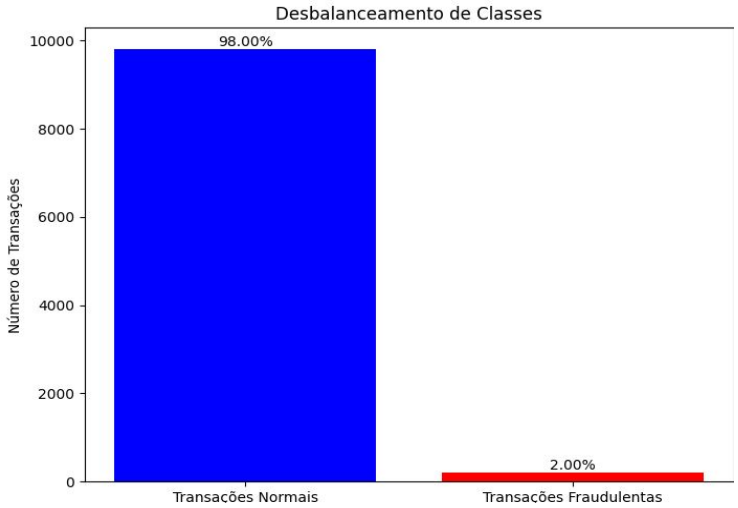
- Os dados desbalanceados ocorrem quando a classe dependente possuem uma grande desproporção, o que pode causar problemas na acurácia do modelo.
- A performance de um modelo é mais seriamente afetada com classes desbalanceadas além da proporção de 90/10.
- O desbalanceamento do conjunto de dados pode influenciar o treinamento do modelo e resultar em falhas na predição.

Dados desbalanceados

- Exemplo.: Em um conjunto de dados de transações bancárias, onde o objetivo é classificar o atributo alvo, em transações normais ou transações fraudulentas.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   valor_transacao  10000 non-null  float64
 1   data_hora        10000 non-null  float64
 2   target           10000 non-null  int64
```

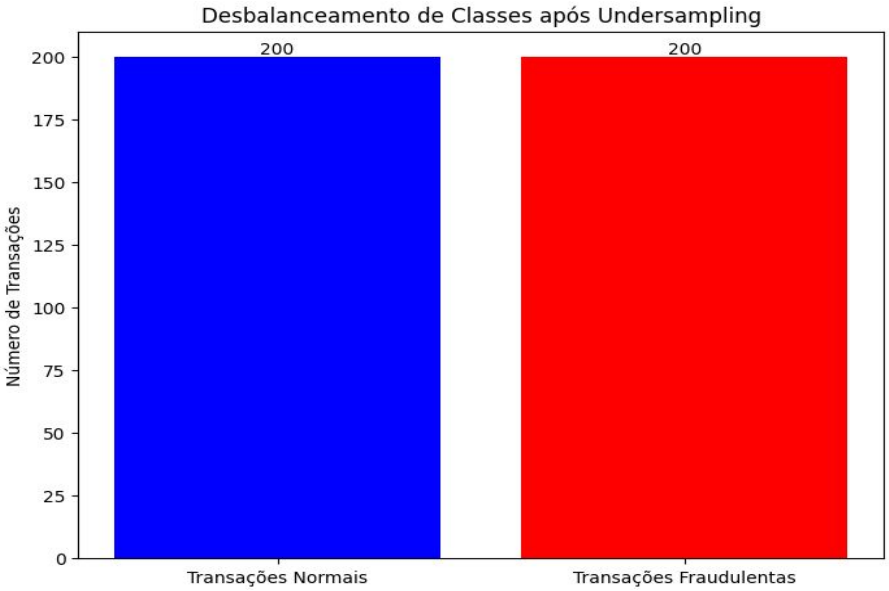
	valor_transacao	data_hora	target
0	898.598684	507.820675	0
1	482.875216	289.579421	0
2	519.987203	57.980832	0
3	514.068963	765.717605	0
4	304.128715	922.014372	0

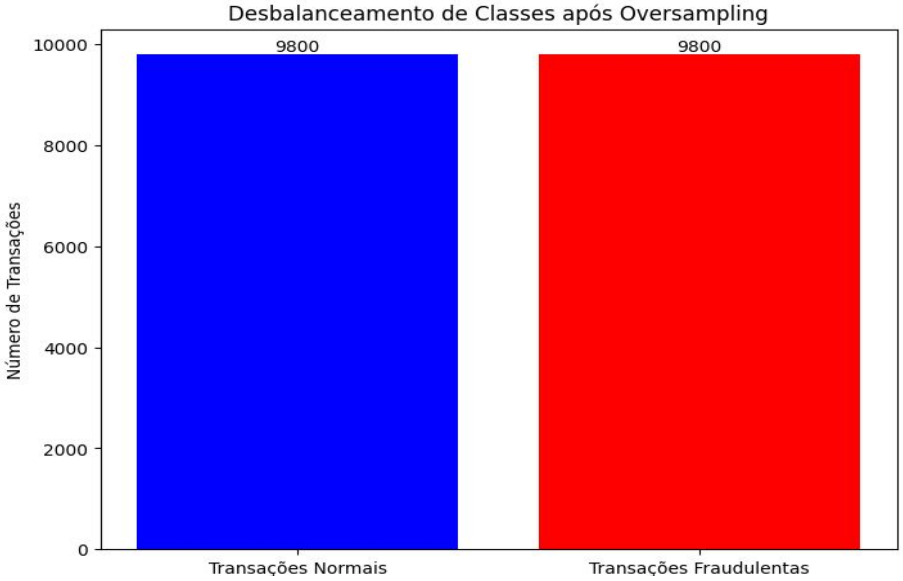


Dados desbalanceados

- Como corrigir o problema dos dados desbalanceados:
  - Coletar mais dados;
  - Undersampling – Reduzir os dados da classe majoritária aleatoriamente de forma que o conjunto de dados fique com uma quantidade equilibrada de registros na classes alvo.
  - Oversampling – Aumentar a quantidade de registros da classe minoritária até que o conjunto de dados fique com uma quantidade equilibrada de registros na classes alvo.

Dados desbalanceados



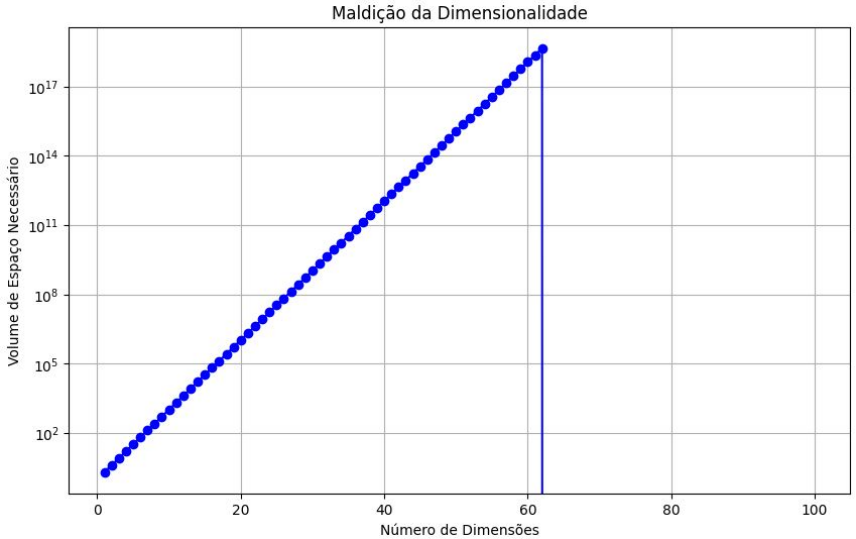


Redução de dimensionalidade

- É o processo de reduzir o número de variáveis aleatórias (colunas – características), a serem inseridas no modelo;
- Necessário quando existem um número muito grande características, que aumenta a complexidade do modelo e dificulta o treinamento;
- Neste caso, é necessário identificar as componentes principais do conjunto de variáveis aleatórias, selecionando as que não sejam linearmente correlacionadas.
    - Ex.: Nº de treinos por dia x Nº de treinos por mês
    - Neste caso uma das duas colunas pode ser eliminada do conjunto de entrada;

Redução de dimensionalidade

- O volume do espaço de dados aumenta exponencialmente com o número de dimensões (ou características). Isso torna a análise de dados mais complexa e menos eficiente.



Transformação de dados

- A transformação de dados tem como objetivo evitar que o algoritmo fique enviesado para as variáveis com maior ordem de grandeza.
- Pode ser aplicado usando duas técnicas:
  - Normalização (Re-escalar);
  - Padronização (z-score normalization).

Transformação de dados

- Normalização (Re-escalar):
  - Converte os valores para um intervalo de 0 e 1 e caso tenha resultado negativo de -1 e 1;

$$X' = \frac{(X - min_X)}{(max_X - min_X)}$$

Transformação de dados

- Normalização (Re-escalar):

```python
import pandas as pd

# Criando um DataFrame de exemplo com valores extremos
data = {
    'Feature1': [10, 20, 30, 40, 50, 1000],  # Valor extremo: 1000
    'Feature2': [1, 2, 3, 4, 5, 100]       # Valor extremo: 100
}

df = pd.DataFrame(data)

# Função para normalizar min-max
def min_max_normalize(column):
    return (column - column.min()) / (column.max() - column.min())

# Aplicando a normalização min-max a cada coluna do DataFrame
df_normalized = df.apply(min_max_normalize)

print("DataFrame original:")
print(df)

print("\nDataFrame normalizado (min-max):")
print(df_normalized)
```

```
DataFrame original:
   Feature1  Feature2
0        10         1
1        20         2
2        30         3
3        40         4
4        50         5
5      1000       100

DataFrame normalizado (min-max):
   Feature1  Feature2
0  0.000000  0.000000
1  0.010101  0.010101
2  0.020202  0.020202
3  0.030303  0.030303
4  0.040404  0.040404
5  1.000000  1.000000
```

Transformação de dados

- Padronização (z-score normalization):
  - Transforma a variável em uma média igual a 0 e um desvio padrão igual a 1.

$$X' = \frac{(\bar{X} - X)}{S_X}$$

Transformação de dados

- Padronização (z-score normalization):

```python
import pandas as pd

# Criando um DataFrame de exemplo com valores extremos
data = {
    'Feature1': [10, 20, 30, 40, 50, 1000],  # Valor extremo: 1000
    'Feature2': [1, 2, 3, 4, 5, 100]       # Valor extremo: 100
}

df = pd.DataFrame(data)

# Função para padronizar (z-score normalization)
def z_score_normalize(column):
    return (column - column.mean()) / column.std()

# Aplicando a padronização a cada coluna do DataFrame
df_standardized = df.apply(z_score_normalize)

print("DataFrame original:")
print(df)

print("\nDataFrame padronizado (z-score):")
print(df_standardized)
```

```
DataFrame original:
   Feature1  Feature2
0        10         1
1        20         2
2        30         3
3        40         4
4        50         5
5      1000       100

DataFrame padronizado (z-score):
   Feature1  Feature2
0 -0.458461 -0.458461
1 -0.433225 -0.433225
2 -0.407988 -0.407988
3 -0.382752 -0.382752
4 -0.357515 -0.357515
5  2.039941  2.039941
```

Pré-processamento

- Base de dados da Iris

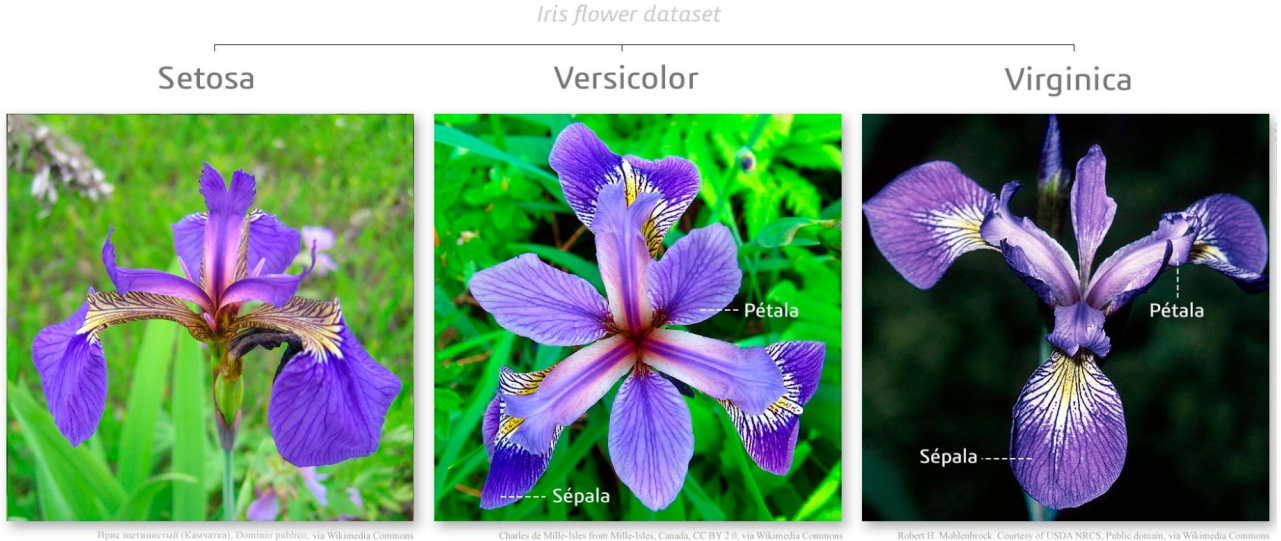
Variables Table

Variable Name	Role	Type	Description	Units	Missing Values
sepal length	Feature	Continuous		cm	no
sepal width	Feature	Continuous		cm	no
petal length	Feature	Continuous		cm	no
petal width	Feature	Continuous		cm	no
class	Target	Categorical	class of iris plant: Iris Setosa, Iris Versicolour, or Iris Virginica		no

Fonte:https://archive.ics.uci.edu/dataset/53/iris

Pré-processamento

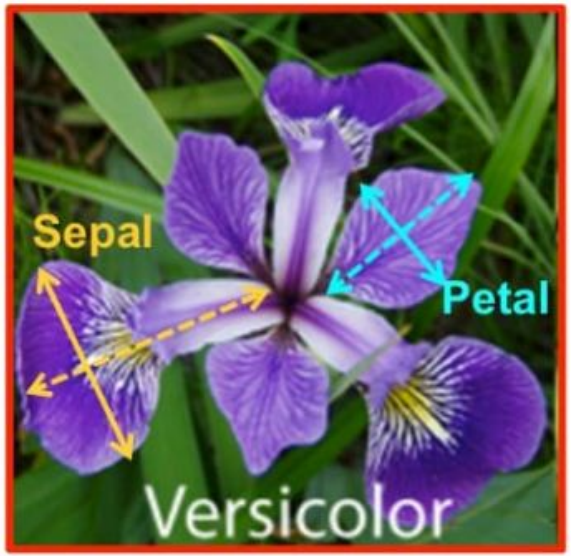
- Base de dados da Iris



Fonte: https://commons.wikimedia.org/wiki/File:Flores_de_%C3%8Dris.png

Pré-processamento

- Base de dados da Iris

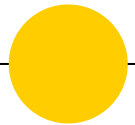


Fonte: https://www.snaplogic.com/machine-learning-showcase/iris-flower-classification

Pré-processamento

- Exercício: Efetuar o tratamento do Dataset da Iris modificado, aplicando as técnicas e funções aprendidas na aula:
- Exibir as 10 primeiras linhas;
- Contar o número de campos com dados nulos;
- Exibir as medidas descritivas;
- Tratar os valores ausentes;
- Remover duplicados e outliers;
- Exibir o gráfico boxplot do antes e após as mudanças;
- Exibir os dados modificados e contar os dados nulos.
- Arquivos: iris_modificado.csv

Dúvidas?
Comentários?



Este material foi desenvolvido com referência e inspiração no conteúdo do **Prof. Edmilson dos Santos de Jesus**, a quem agradeço pela colaboração e compartilhamento.