revisao

November 25, 2025

1 Revisão Machine Learning

1.1 Importação de arquivos

[15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('datasetaqui.csv')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[15], line 6
      3 import matplotlib.pyplot as plt
      4 import seaborn as sns
----> 6 df = pd.read_csv('datasetaqui.csv')

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\io\parsers\readers
 ↪py:1026, in read_csv(filepath_or_buffer, sep, delimiter, header, names,␣
 ↪index_col, usecols, dtype, engine, converters, true_values, false_values,␣
 ↪skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na,␣
 ↪na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format,␣
 ↪keep_date_col, date_parser, date_format, dayfirst, cache_dates, iterator,␣
 ↪chunksize, compression, thousands, decimal, lineterminator, quotechar,␣
 ↪quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect␣
 ↪on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision,␣
 ↪storage_options, dtype_backend)
  1013 kwds_defaults = _refine_defaults_read(
  1014     dialect,
  1015     delimiter,
  (…)
  1022     dtype_backend=dtype_backend,
  1023 )
  1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\io\parsers\readers
 ↪py:620, in _read(filepath_or_buffer, kwds)
   617 _validate_names(kwds.get("names", None))
```

```
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\io\parsers\readers
 ↪py:1620, in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\io\parsers\readers
 ↪py:1880, in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File ~\AppData\Roaming\Python\Python312\site-packages\pandas\io\common.py:873,␣
 ↪in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text,␣
 ↪errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline="",
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'datasetaqui.csv'
```

[ ]:

1.2 Padronização dos Dados

1.2.1 Standard Scaler

Responsável por deixar todos os dados em uma mesma escala. * Centraliza a média em 0 * Ajusta o desvio padrão para 1

É necessário usá-lo em modelos que usem **regularização**, visto que eles podem interpretar dados maiores como mais importantes, o que nem sempre é verdade.

$$\frac{x_i - \mu(x)}{\sigma} \qquad \frac{x_i - \mu(x)}{\sigma}$$

Fórmula

Código

[13]:
```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train) # aprende média e desvio nos dados
x_test = scaler.transform(x_test) # aplica
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[13], line 4
      1 from sklearn.preprocessing import StandardScaler
      3 scaler = StandardScaler()
----> 4 x_train = scaler.fit_transform(x_train) # aprende média e desvio nos␣
 ↪dados
      5 x_test = scaler.transform(x_test) # aplica

NameError: name 'x_train' is not defined
```

[ ]:

1.3 Encoder

1.3.1 Label Encoding

Cada classe terá seu rótulo transformado para um valor numérico

- É comumente utilizado em algoritmos de **Classificação**

- É utilizado em dados ordenáveis, como:
  - Bronze, prata e ouro;
  - Criança, adulto e idoso;
  - Baixo, médio e alto.
- É preciso criar um objeto LabelEncoder **para cada coluna**

Color	Size	Price
Red	Small	10
Green	Medium	20
Blue	Large	30
Red	Large	25
Green	Small	15

Category	Encoded Value
Color: Red	0
Color: Green	1
Color: Blue	2

Category	Encoded Value
Size: Small	0
Size: Medium	1
Size: Large	2

Color_Encoded	Size_Encoded	Price
0	0	10
1	1	20
2	2	30
0	2	25
1	0	15

Código

```
[ ]: from sklearn.preprocessing import LabelEncoder

encoding_col1 = LabelEncoder()
endoding_col2 = LabelEncoder()

x[:,1] = encoding_col1.fit_transform(x[:,1])
x[:,2] = encoding_col2.fit_transform(x[:,2])
```

1.3.2 One-hot Encoding

Cada categoria é transformada em um atributo: dummy variable, um valor binário que informa a ocorrência

Quando utilizar?

- Quando a variável categórica **não tem ordem** (nominal);
- Quando o número de categorias não é muito **grande**;

Observações

- Muitas colunas podem gerar um espaço de características de alta dimensão, que pode causar super ajuste e ter um custo computacional muito alto.

- Maldição da Dimensionaldiade: Dados esparços, muitas colunas com valor zero, tornando difícil encontrar valores nos dados
- Dummy Variable Trap: valores de colunas binárias podem ser previstos a partir dos valores de outras colunas.

Color	Size	Price
Red	Small	10
Green	Medium	20
Blue	Large	30
Red	Large	25
Green	Small	15

Color_Red	Color_Green	Color_Blue	Size_Small	Size_Medium	Size_Large	Price
1	0	0	1	0	0	10
0	1	0	0	1	0	20
0	0	1	0	0	1	30
1	0	0	0	0	1	25
0	1	0	1	0	0	15

Código

```
[ ]: from sklearn.preprocessing import OneHotEncoder
     from sklearn.compose mport ColumnTransformer

     onehotX = ColumnTransformer(transformers=[('OneHot',␣
      ↪OneHotEncoder(handle_unknown='ignore', [6,7], remainder='passthrough')
     x = onehotX.fit_transform(x)
```

```
[ ]:
```

1.4 Regressão

- **Não** possui hiperparâmetro
- **Métrica de erro:** MAE
- **Métrica de desempenho:** Score

MAE Mean Absolute Error

```
[ ]: from sklearn.metrics import mean_absolute_error

     mae = mean_absolute_error(y_test, prev)
```

Score

```
[11]: regressor.score(x_train, y_train)
      regressor.score(x_test, y_test)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 1
----> 1 regressor.score(x_train, y_train)
      2 regressor.score(x_test, y_test)

NameError: name 'x_train' is not defined
```

1.4.1 Regressão Linear Simples

Modelagem da relação entre variáveis numéricas (variável dependente y e variáveis explanatórias x)

Intersecção O ponto de encontro da linha no eixo Y, onde X = 0

Inclinação Fator que determina a inclinação da linha onde a cada unidade que aumenta a variável **independente**(x), a variável de **resposta**(y) sobe o valor da inclinação

Fórmula

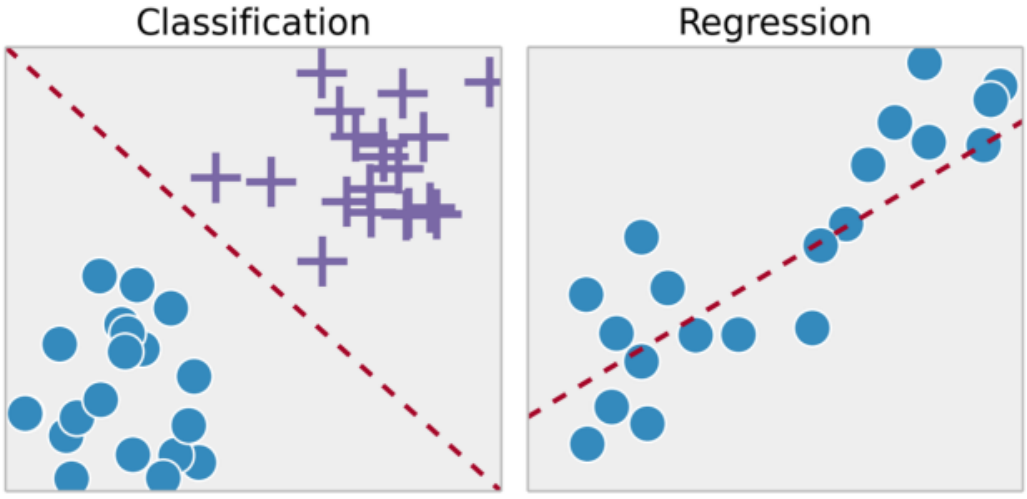
P = b + m * v onde:

p: previsão

b(constante): intersecção

m(coeficiente): inclinação

v: valor a ser previsto em x



[3]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x, y)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[3], line 3
      1 from sklearn.linear_model import LinearRegression
      2 regressor = LinearRegression()
----> 3 regressor.fit(x, y)

NameError: name 'x' is not defined
```

1.4.2 Regressão Linear Múltipla

É similar a Regressão Linear **Simples**, porém mais complexa.

- Possui **duas ou mais** variáveis exploratórias

Fórmula $P = b + m1 * v1 + m2 * v2 + ... + mn * vm$

Código

[ ]:
```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(x_train, y_train)
prev = regressor.predict(x_test)
```

1.4.3 Regressão Linear Polinomial

- Usada quando a relação entre x e y é **curva**, e não pode ser representada apenas por uma linha reta.
- Quando falamos em código, falamos em algo basicamente **igual** a uma regressão linear simples.
- É preciso adequar os dados e criar novas features (elevadas a n) para finalmente treiná-lo

Fórmula $P = C + m1 * v1 + m2 * v2 + ... + mn * v1\hat{}n$

Código

[ ]:
```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 2)
x_train = poly.fit_transform(x_train)
x_test = poly.transform(x_test)

regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

1.4.4 Elastic Net

Utilizado quando temos duas ou mais variáveis exploratórias.

Quando temos muitas variáveis, ou quando elas possuem valores muito parecidos, o modelo pode ficar "confuso" e distribuir pesos muito grandes, ocasionando em **overfitting**.

O Elastic usa:

- Ridge para remover exageros
- Lasso para remover variáveis que não são fortemente relacionadas com y.

$$\min_{w} \sum_{i=1}^{N} (\vec{w} \cdot \vec{x}_i + \beta - y_i)^2 + \alpha\rho|w| + \alpha(1-\rho)w^2$$

Ridge Penaliza colocando um viés que reduz os grandes pesos o máximo possível, mas sem zerar, assim fazendo com que a variável contribua menos para a predição

$$\min_{w} \sum_{i=1}^{N} (\vec{w} \cdot \vec{x}_i + \beta - y_i)^2 + \alpha w^2$$

Lasso

- Penaliza o valor, assim como o Ridge, mas, ao invés de penalizar apenas os pesos de grande valor, ele penaliza os de baixo valor também.
- A penalização ocorre até que o valor seja zero.
- Os atributos zerados são descartados da predição.

$$\min_{w} \sum_{i=1}^{N} (\vec{w} \cdot \vec{x}_i + \beta - y_i)^2 + \alpha|w|$$

Código

[ ]:
```
from sklearn.linear_model import ElasticNet

regressor_en = ElasticNet(alpha=0.1, l1_ratio=0.5, random_state=0)
regressor_en.fit(x_train, y_train)

prev = regressor_en.predict(x_test)
```

[ ]:
```

```

1.5 Classificação

- **Métrica de erro:** Matriz de confusão
- **Métrica de desempenho:** accuracy_score

1.5.1 Matriz de confusão (Confusion Matrix)

[ ]:
```
from yellowbrick.classifier import ConfusionMatrix

plt.figure(figsize=(4,4))
cm = ConfusionMatrix(svm)
cm.fit(x_train, y_train)
cm.score(x_test, y_test)
```

1.5.2 Accuracy Score

[ ]:
```
from sklearn.metrics import accuracy_score, classification_report
previsao = svm.predict(x_test)
accuracy_score(y_test, previsao)
```

1.5.3 Naive Bayes

É um algoritmo de classificação baseado no Teorema de Bayes.