Machine Learning

Introduction to Machine Learning with Python.

IA - Quando um computador realiza qualquer ação humana

- tarefa - T
- experiência - E
- performance - P

1) Uma tarefa específica: Prever o resultado de um jogo de futebol

2) Fornecer dados à máquina

3) Análise de padrões

Nenhuma previsão é 100%

Aprendizado Supervisionado

 Quando tentamos prever uma variável dependente a partir de uma lista de variáveis independentes

Var Independentes	Var Dependentes
Anos de carreira	Salário
Idade do carro	Risco de Acidente
Temperatura	Venda de Sorvete

Google Colab

>> Edição e Execução de Programas em Python

Pandas: é uma das bibliotecas mais usadas em Machine Learning

- Análise de Dados
- Iniciar Análise de Dados
- Permite Ler Manipular

Scikit-learn: Aplicar algoritmos de classificação, regressão, agrupamento..

Antes de executar aprendizado de máquina usa-se agoritmos de preparação de dados

Algoritmo Naive Bayes

Aprende para classificar

“Naive” Desconsidera a correlação entre as variáveis

Diagnóstico de Doenças

▼ Sobre

Naive Bayes
é **um algoritmo que gera uma tabela de probabilidades a partir de uma técnica de classificação de dados.**

É usado para o machine learning, mas a técnica é famosa no meio acadêmico da estatística. Seu racional é baseado nos estudos de Thomas Bayes e “naive” significa ingênuo

Coleta de dados 🌷🌷🌷

Machine Learning Repository

```
import pandas as pd
base = pd.read_csv("/content/iris.data")

base
```

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
```

pandas.DataFrame.iloc - pandas 1.5.1 documentation

Purely integer-location based indexing for selection by position. is primarily integer position based (from to of the axis), but may also be used with a boolean array. Allowed inputs are: A boolean array. A function with one argument (the calling Series or DataFrame) and that returns valid output for indexing (one of the above).

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.iloc.html?highlight=iloc#pandas.DataFrame.iloc

LabelEncoder → Transforma dados (labels) em dados numéricos

```
from sklearn.preprocessing import LabelEncoder

labelencoder_classe = LabelEncoder()
classe = labelencoder_classe.fit_transform(classe)

classe
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Escalonamento dos atributos (padronização)

Pré-processamento - sklearn

Assertividade maior com a padronização

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

previsores = scaler.fit_transform(previsores)

previsores
```

```
array([[-1.1483555 , -0.11805969, -1.35396443, -1.32506301],
       [-1.3905423 ,  0.34485856, -1.41098555, -1.32506301],
       [-1.51163569,  0.11339944, -1.29694332, -1.32506301],
       [-1.02726211,  1.27069504, -1.35396443, -1.32506301],
       [-0.54288852,  1.9650724 , -1.18290109, -1.0614657 ],
       [-1.51163569,  0.8077768 , -1.35396443, -1.19326436],
       [-1.02726211,  0.8077768 , -1.29694332, -1.32506301],
       [-1.75382249, -0.34951881, -1.35396443, -1.32506301],
       [-1.1483555 ,  0.11339944, -1.29694332, -1.45686167],
       [-0.54288852,  1.50215416, -1.29694332, -1.32506301],
       [-1.2694489 ,  0.8077768 , -1.23992221, -1.32506301],
       [-1.2694489 , -0.11805969, -1.35396443, -1.45686167],
       [-1.87491588, -0.11805969, -1.52502777, -1.45686167],
```

Treinamento dos dados

Pacote → `train_test_split`

```
from sklearn.model_selection import train_test_split

previsores_treinamento, previsores_teste, classe_treinamento, class_test = train_test_split(previsores, classe, test_size=0.25, random
```

> Quanto mais dados usar no treinamento, melhor!

```
from sklearn.naive_bayes import GaussianNB
classificador = GaussianNB()
classificador.fit(previsores_treinamento, classe_treinamento)
previsoes = classificador.previsores(previsores_teste)

previsoes
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
acuracia = accuracy_score(class_test, previsoes)
acuracia
```

```
0.868421052631579
```

```
matriz = confusion_matrix(class_test, previsoes)
matriz
```

```
array([[14,  0,  0],
       [ 0, 13,  1],
       [ 0,  4,  6]])
```
