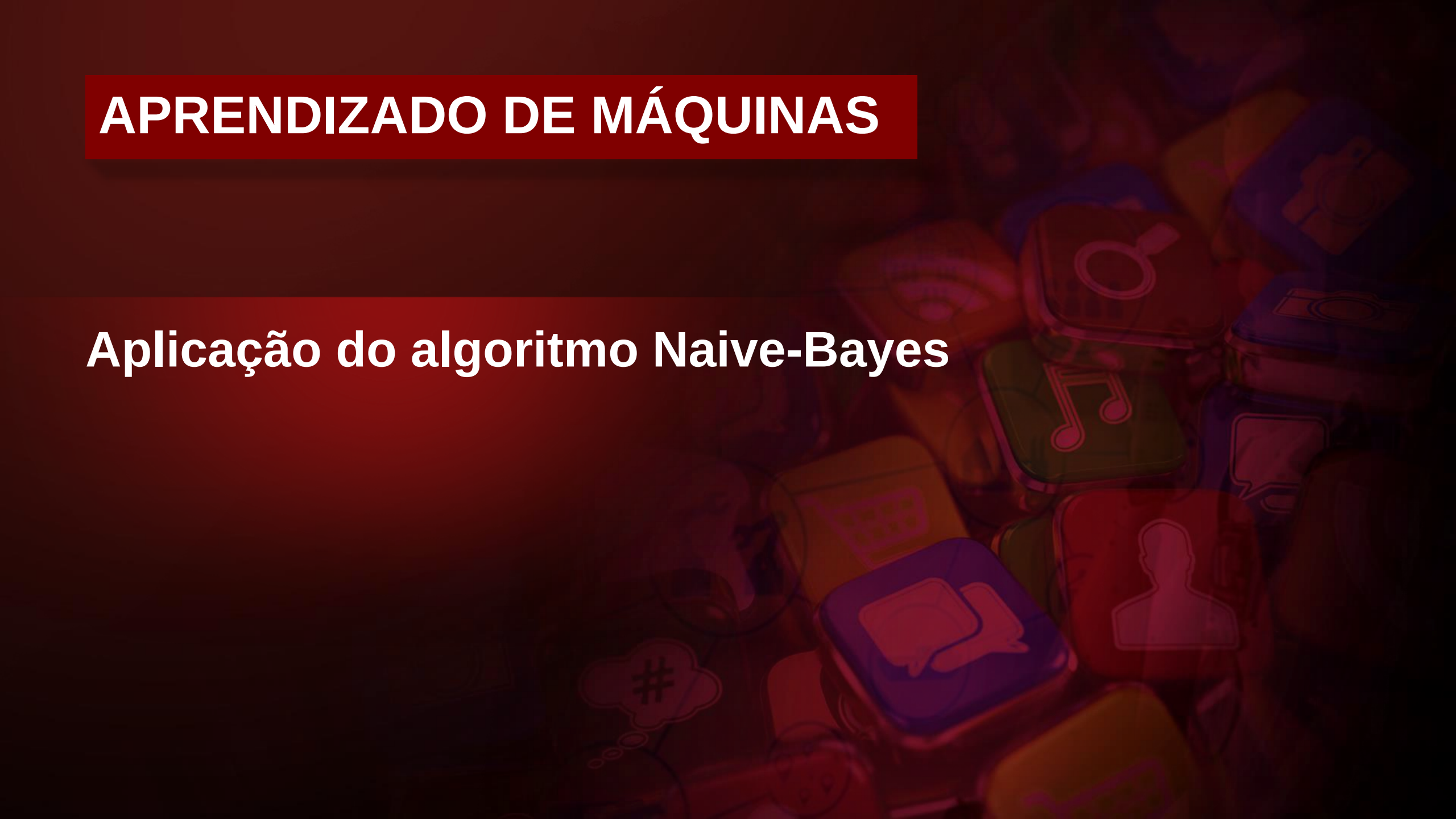
APRENDIZADO DE MÁQUINAS

Aplicação do algoritmo Naive-Bayes



TÓPICOS

1. Classificador Naive-Bayes
2. Conjunto de dados: Iris
3. Código Python
4. Análise do Algoritmo

CLASSIFICAÇÃO NAIVE- BAYES

Hipótese principal: variáveis para identificação de uma dada classe são independentes umas das outras.

Fonte: br.freepik

CONJUNTO DE DADOS UTILIZADO - IRIS

O conjunto de dados de flores de iris é muito famoso e consiste em 3 classes de flores.
Existem 4 variáveis independentes:
sepal_length **,** ***sepal_width*** **,** ***petal_length*** **e** ***petal_width*** **.**
A variável dependente é a ***espécie*** **que vamos prever usando as quatro características independentes das flores.**
Existem 3 classes de espécies: setosa, versicolor e virginica.

Foto de Annie Spratt no Unsplash

ALGORITMO NAIVE-BAYES - PYTHON

jupyter Navi-Bayes-Aula Last Checkpoint: 22 minutos atrás (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [15]:
```
import random
random.seed(42) # define the seed (important to reproduce the results)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [16]:
```
data = pd.read_csv('Iris.csv', header=(0))
data = data.dropna(axis='rows') #remove NaN
```

ALGORITMO NAIVE-BAYES - PYTHON

In [17]:
```
# armazena os nomes das classes
classes = np.array(pd.unique(data[data.columns[-1]]), dtype=str)
print("Número de linhas e colunas na matriz de atributos:", data.shape)
attributes = list(data.columns)
data.head(10)
```

```
Número de linhas e colunas na matriz de atributos: (150, 6)
```

Out[17]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

ALGORITMO NAIVE-BAYES - PYTHON

In [18]:
```
data = data.to_numpy()
nrow,ncol = data.shape
y = data[:,-1]
X = data[:,0:ncol-1]
```

In [19]:
```
# Selecionando os conjuntos de treinamento e teste.
from sklearn.model_selection import train_test_split
p = 0.7 # fracao de elementos no conjunto de treinamento
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = p, random_state = 42)
```

In [20]:
```
#definição de uma função para calcular a densidade de probabilidade conjunta
#definição de função de verossimilhança
def likelyhood(y, Z):
    def gaussian(x, mu, sig):
        return np.exp(-np.power(x - mu, 2.) / (2 * np.power(sig, 2.)))
    prob = 1
    for j in np.arange(0, Z.shape[1]):
        m = np.mean(Z[:,j])
        s = np.std(Z[:,j])
        prob = prob*gaussian(y[j], m, s)
    return prob
```

ALGORITMO NAIVE-BAYES - PYTHON

In [21]:
```
#cálculo da estimação para cada classe:
P = pd.DataFrame(data=np.zeros((X_test.shape[0], len(classes))), columns = classes)
for i in np.arange(0, len(classes)):
    elements = tuple(np.where(y_train == classes[i]))
    Z = X_train[elements,:][0]
    for j in np.arange(0,X_test.shape[0]):
        x = X_test[j,:]
        pj = likelyhood(x,Z)
        P[classes[i]][j] = pj*len(elements)/X_train.shape[0]
```

ALGORITMO NAIVE-BAYES - PYTHON

In [22]:
```
#imprime a probabilidade pertencer a cada classe de acordo com as observações no conjunto de teste :
P.head(10)
```

Out[22]:

	Iris-setosa	Iris-versicolor	Iris-virginica
0	2.203966e-92	4.345947e-03	8.401969e-08
1	1.479191e-04	7.322596e-20	5.675329e-35
2	3.574579e-294	6.684121e-19	2.223171e-05
3	7.304744e-96	4.012212e-03	1.424931e-06
4	1.666488e-108	8.057105e-04	1.410390e-06
5	1.405723e-03	6.169399e-17	9.013438e-32
6	8.718053e-55	2.340174e-03	4.316097e-11
7	2.564878e-183	6.862286e-14	1.482946e-03
8	9.841417e-98	7.266205e-04	6.224668e-09
9	1.520404e-62	6.288432e-03	1.024099e-08

ALGORITMO NAIVE-BAYES - PYTHON

In [14]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
img=mpimg.imread('iris_types.jpg')
plt.figure(figsize=(20,40))
plt.axis('off')
plt.imshow(img)
```

Out[14]:
```
<matplotlib.image.AxesImage at 0x1ee28beadf0>
```

ALGORITMO NAIVE-BAYES - PYTHON

In [24]:

```
#calcula a acurácia
from sklearn.metrics import accuracy_score

y_pred = []
for i in np.arange(0, P.shape[0]):
    c = np.argmax(np.array(P.iloc[[i]]))
    y_pred.append(P.columns[c])
y_pred = np.array(y_pred, dtype=str)

score = accuracy_score(y_pred, y_test)
print('Accuracy:', score)
```

```
Accuracy: 1.0
```

ALGORITMO NAIVE-BAYES - PYTHON

```
In [26]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test, y_pred)
         from sklearn.metrics import accuracy_score
         print ("Accuracy : ", accuracy_score(y_test, y_pred))
         cm

         Accuracy :  1.0

Out[26]: array([[19,  0,  0],
                [ 0, 13,  0],
                [ 0,  0, 13]], dtype=int64)
```

		Valor Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte: br.freepik

ALGORITMO NAIVE-BAYES - PYTHON

In [25]:
```
#classificação usando a biblioteca do scikit-learn
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
score = accuracy_score(y_pred, y_test)
print('Accuracy:', score)
```

```
Accuracy: 1.0
```

ALGORITMO NAIVE-BAYES - PYTHON

In [12]:
```
#Outra maneira de efetuarmos a classificação é assumirmos que os atributos possuem distribuição diferente da normal.
#Uma possibilidade é assumirmos que os dados possuem distribuição de Bernoulli.
from sklearn.naive_bayes import BernoulliNB

model = BernoulliNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
score = accuracy_score(y_pred, y_test)
print('Accuracy:', score)
```

```
Accuracy: 0.28888888888888886
```

ALGORITMO NAIVE-BAYES - PYTHON

In [27]:
```
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
df
```

Out[27]:

	Real Values	Predicted Values
0	Iris-versicolor	Iris-versicolor
1	Iris-setosa	Iris-setosa
2	Iris-virginica	Iris-virginica
3	Iris-versicolor	Iris-versicolor
4	Iris-versicolor	Iris-versicolor
5	Iris-setosa	Iris-setosa
6	Iris-versicolor	Iris-versicolor
7	Iris-virginica	Iris-virginica
8	Iris-versicolor	Iris-versicolor
9	Iris-versicolor	Iris-versicolor
10	Iris-virginica	Iris-virginica
11	Iris-setosa	Iris-setosa
12	Iris-setosa	Iris-setosa
13	Iris-setosa	Iris-setosa
14	Iris-setosa	Iris-setosa
15	Iris-versicolor	Iris-versicolor

APRENDIZADO DE MÁQUINAS

Aplicação do algoritmo Naive-Bayes