*Por Leonardo Damasio*

**Machine Learning Tutorial**

**DIVIDINDO BASE EM TREINO E TESTE**

**1º Passo** – Importar arquivo

Considerando que seja um arquivo CSV, no qual o separador é feito por vírgulas

**import** **pandas** **as** **pd**

arquivo = pd.read\_csv("caminho/arquivo.csv")

**2º Passo** – Entender disposição (linhas, colunas) da tabela

arquivo.shape()

**3º Passo** – Dividir dados de entrada (x) e saída (y)

Considerando que o arquivo tenha 20 colunas (variáveis) de entrada, começando na indexação [0] e terminando na [19]

x = arquivo.iloc[:,0:20].values

y = arquivo.iloc[:,20].values

**4º Passo** – Converter campos *string* em *int* através da conversão de vetores

Considerando que as colunas de indexação [0], [2], [3], [5], [6], [8], [9], [11], [13], [14], [16], [18] e [19] contenham *strings*

**from** **sklearn.preprocessing** **import** LabelEncoder

labelencoder = LabelEncoder()

transform = [0,2,3,5,6,8,9,11,13,14,16,18,19]

**for** i **in** transform:

x[:,i] = x.fit\_transform(x[:,i])

**5º Passo** – Dividir dados de entrada (x) e saída (y) em treino e teste

Considerando que seja feita uma divisão 70% treino e 30% teste

**from** **sklearn.model\_selection** **import** train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, random\_state = 0)

**MODELOS DE CLASSIFICAÇÃO**

* **Naive Bayes (GaussianNB)**

**from** **sklearn.naive\_bayes** **import** GaussianNB

naive\_bayes = GaussianNB()

naive\_bayes.fit(x\_train, y\_train)

predicao\_y\_test = naive\_bayes.predict(x\_test)

* **Árvores de decisão (DecisionTreeClassifier)**

**from** **sklearn.tree** **import** DecisionTreeClassifier

arvore = DecisionTreeClassifier()

arvore.fit(x\_train, y\_train)

predicao\_y\_test = arvore.predict(x\_test)

**import** **graphviz**

**from** **sklearn.tree** **import** export\_graphviz

export\_graphviz(arvore, out\_file = "tree.dot")

* **Support Vector Machine (SVM/SVC)**

**from** **sklearn.svm** **import** SVC

svm = SVC()

svm.fit(x\_train, y\_train)

predicao\_y\_test = svm.predict(x\_test)

* **KNeighborsClassifier (KNN)**

**from** **sklearn.neighbors** **import** KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors = 3)

knn.fit(x\_train, y\_train)

predicao\_y\_test = knn.predict(x\_test)

* **RandomForestClassifier**

**from** **sklearn.ensemble** **import** RandomForestClassifier

floresta = RandomForestClassifier(n\_estimators = 100)

floresta.fit(x\_train, y\_train)

predicao\_y\_test = floresta.predict(x\_test)

**VERIFICAÇÃO DE RESULTADOS**

* Verificar porcentagem de acertos

**from** **sklearn.metrics** **import** accuracy\_score

taxa\_acerto = accuracy\_score(y\_test, predicao\_y\_test)

taxa\_acerto

* Descobrir importâncias das variáveis

**from** **sklearn.ensemble** **import** ExtraTreesClassifier

forest = ExtraTreesClassifier()

forest.fit(x\_train, y\_train)

importancias = forest.feature\_importances\_

importancias

* Gerar matriz de confusão

**from** **sklearn.metrics** **import** confusion\_matrix

matriz = confusion\_matrix(y\_test, predicao\_y\_test)

matriz

* Plotar matriz de confusão

**from** **yellowbrick.classifier** **import** ConfusionMatrix

matriz = ConfusionMatrix(GaussianNB())

matriz.fit(x\_train, y\_train)

matriz.score(x\_test, y\_test)

matriz.poof()

* Validação de novas entradas

novo\_arquivo = pd.read\_csv("novo\_arquivo.csv")

novo\_arquivo = novo\_arquivo.iloc[:,0:20].values

**for** i **in** transform:

novo\_arquivo[:,i] = labelencoder.fit\_transform(novo\_arquivo[:,i])

predicao = naive\_bayes.predict(novo\_arquivo)

print(predicao)