Machine Learning Tutorial

Por Leonardo Damasio

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_s
tate = 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)
```