UNIVERSIDADE EVANGÉLICA DE GOIÁS – UNIEVANGÉLICA

ENGENHARIA DE SOFTWARE

Camila Souza

Bianca Abreu Félix

Felipe Franco

Algoritmos de Classificação

Anápolis - GO

Maio, 2022

Link Github: https://github.com/FelpsFranco/Algoritmo_De_Classificacao

Dados: Presença de Doenças Cardíacas

Introdução

Algoritmos de Classificação ou Algoritmos de Machine Learning podem ser classificados em Aprendizagem Supervisionada, Aprendizagem Não supervisionada e Aprendizagem por Reforço. O uso desses algoritmos dependem do tipo de dado que será tratado e quais respostas precisam ser obtidas com sua implementação.

Para sua utilização é primeiramente necessário a extração de características, coleta de dados que serão utilizados. Após a coleta, será utilizado uma nova amostra de dados na qual ainda não temos informações e nem características, inserindo esses dados no modelo aprendido.

Alguns exemplos de algoritmos de classificação:

. **K-NN (Nearest Neighbors) ou K-vizinhos mais próximos:** ele se baseia na distância entre pontos para classificação dos dados. Para este modelo você precisa escolher um número K e averiguar qual classe mais se repete entre os K vizinhos mais próximos. A arbitrariedade na escolha do valor K é uma desvantagem deste método de classificação.

. **SVM (Support Vector Machine):** esse algoritmo de classificação binária, é utilizado em problemas no quais existem apenas duas classes, ele propõe traçar uma linha que divide o conjunto de dados em dois.

. **Random Forest (Florestas de decisão aleatória):** é um método que opera construindo uma infinidade de árvores de decisão em tempo de treinamento. A saída da floresta aleatória é a classe selecionada pela maioria das árvores.

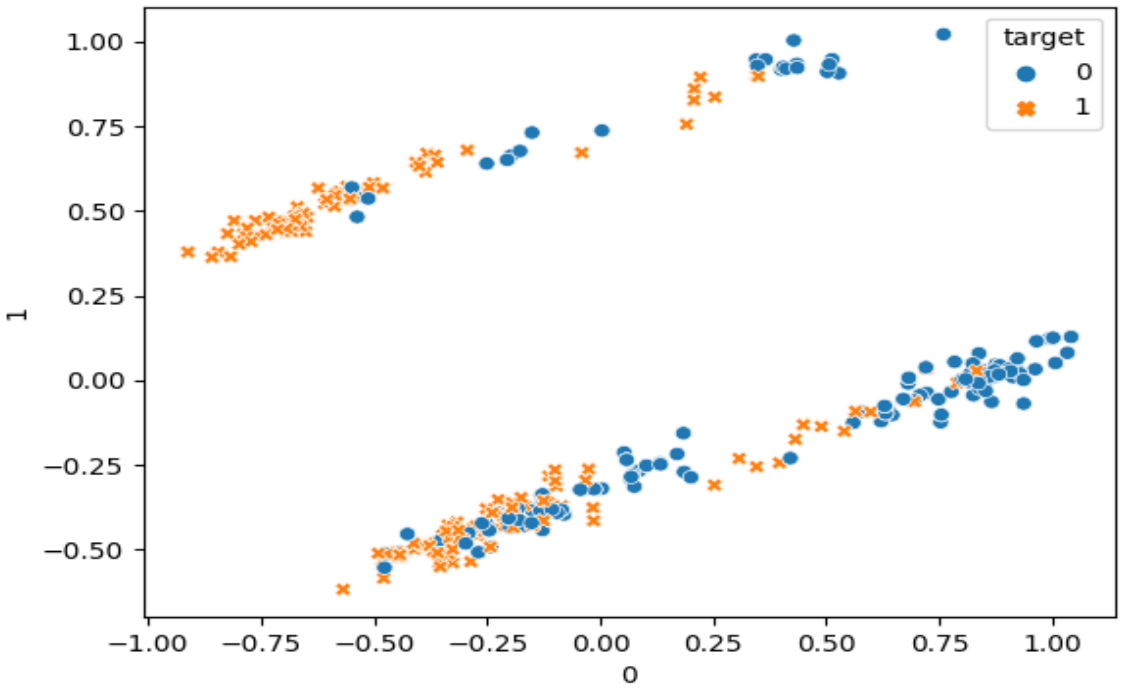
Desenvolvimento:

Bibliotecas utilizadas: pandas, sklearn, seaborn, matplotlib, sklearn.decomposition, sklean.metrics, sklearn.model_selection, sklearn.neighbors, slearn.svm, sklearn.ensemble.

```
import pandas as pd
from sklearn import preprocessing
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, precision_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

Antes a aplicação das formas de classificação foi realizado um tratamento de dados, carregando o Excel (.csv) e realizando uma normalização dos dados (Pré-Processamento) foi utilizado o transformado MinMaxScaler onde recursos são dimensionados para um determinado intervalo.

Feito a normalização de dados, realizamos a segmentação dos dados em Classes, para possibilitar sua visualização foi reduzido a dimensionalidade dos dados e registrado os novos valores. A partir da biblioteca matplotlib plotamos o seguinte dado colorindo pelas classes.



Após o gráfico é feito a separação de treinamento e testes, obtendo o resultado:

Treinamento: 717
Teste: 308

K-NN (Nearest Neighbors)

Aplicando K-NN, instauramos um modelo de dados utilizando primeiramente KNeighborsClassifier (**n_neighbors = 2)** conseguimos obter:

```
K-NN

Acurácia K-NN: 94 %
Precisão K-NN: 88 %
F1-Score K-NN: 94 %
```

O K-NN conseguiu classificar satisfatoriamente o conjuntos de dados.

Porém, realizando um novo teste, aumentando a quantidade de amostras para classificação KNeighborsClassifier (**n_neighbors = 3)**

```
K-NN

Acurácia K-NN: 89 %
Precisão K-NN: 88 %
F1-Score K-NN: 89 %
```

Conseguimos a mesma precisão porém perdemos acurácia.

Algoritmo Utilizado:

```
40  # ------------------------------------------------------------------------ #
41  # K-NN (Nearest Neighbors)
42
43  model1 = KNeighborsClassifier(n_neighbors=3)
44  model1.fit(X_train, y_train)
45
46  y_pred = model1.predict(X_test)
47  acc = accuracy_score(y_pred, y_test)
48  pre = precision_score(y_pred, y_test)
49  f1 = f1_score(y_pred, y_test)
50
51  print('\n\nK-NN')
52  print('\nAcurácia K-NN:', int(acc * 100), '%')
53  print('Precisão K-NN:', int(pre * 100), '%')
54  print('F1-Score K-NN:', int(f1 * 100), '%')
```

SVM (Support Vector Machines)

O modelo de SVM foi utilizado para termos de comparação com o K-NN, criando um segundo modelo de dados para aplicação. Novamente tratando Acurácia, Precisão e F1_Score.

Obtivemos o seguinte resultado:

```
SVM

Acurácia SVM: 85 %
Precisão SVM: 82 %
F1-Score SVM: 85 %
```

Conseguimos visualizar que o SVM obteve resultados piores que o K-NN ficando atrás em todos os pontos levantados (Para esse tipo de dados).

Algoritmo Utilizado:

```
57  # ------------------------------------------------------------------- #
58  # SVM (Support Vector Machines)
59  model2 = SVC()
60  model2.fit(X_train, y_train)
61  y_pred = model2.predict(X_test)
62
63  acc_svm = accuracy_score(y_pred, y_test)
64  pre_svm = precision_score(y_pred, y_test)
65  f1_svm = f1_score(y_pred, y_test)
66
67  print('\n\nSVM')
68  print('\nAcurácia SVM:', int(acc_svm * 100), '%')
69  print('Precisão SVM:', int(pre_svm * 100), '%')
70  print('F1-Score SVM:', int(f1_svm * 100), '%')
```

Random Forest

Realizando mais uma comparação em termos de classificação, o Random Forest foram utilizados o parâmetros **RandomForestClassifier (n_estimators=100, random_state=26)**.

Obtivemos o seguinte resultado:

```
Random Forest

Acurácia Random Forest: 99 %
Precisão Random Forest: 98 %
F1-Score Random Forest: 99 %
```

Com este resultado, o Random Forest fica à frente do K-NN e SVM para estes dados analisados.

Algoritmo Utilizado:

```
# ------------------------------------------------------------------- #
# Random Forest
model3 = RandomForestClassifier(n_estimators=100, random_state=26)
model3.fit(X_train, y_train)
y_pred = model3.predict(X_test)
acc_random = accuracy_score(y_pred, y_test)
pre_random = precision_score(y_pred, y_test)
f1_random = f1_score(y_pred, y_test)

print('\n\nRandom Forest')
print('\nAcurácia Random Forest:', int(acc_random * 100), '%')
print('Precisão Random Forest:', int(pre_random * 100), '%')
print('F1-Score Random Forest:', int(f1_random * 100), '%')
```