K Nearest Neighbors - Resolvido

September 15, 2025

1 K Nearest Neighbors - Projeto

Bem-vindo ao Projeto de KNN! Este será um projeto simples e muito parecido com o notebook, com a diferença de que você receberá outro conjunto de dados. Vá em frente e siga as instruções abaixo. ## Importar bibliotecas **Importe pandas, seaborn e as bibliotecas usuais.**

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
```

1.1 Obtenha os dados

Leia o arquivo csv 'KNN_Project_Data' em um DataFrame

[2]:
```
dados = pd.read_csv('KNN_Project_Data')
```

Verifique o cabeçalho do DataFrame.

[3]:
```
dados.head()
```

[3]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MGJM	JHZC	TARGET CLASS
0	1636.670614	817.988525	2565.995189	358.347163	550.417491	1618.870897	2147.641254	330.727893	1494.878631	845.136088	0
1	1013.402760	577.587332	2644.141273	280.428203	1161.873391	2084.107872	853.404981	447.157619	1193.032521	861.081809	
2	1300.035501	820.518697	2025.854469	525.562292	922.206261	2552.355407	818.676686	845.491492	1968.367513	1647.186291	
3	1059.347542	1066.866418	612.000041	480.827789	419.467495	685.666983	852.867810	341.664784	1154.391368	1450.935357	
4	1018.340526	1313.679056	950.622661	724.742174	843.065903	1370.554164	905.469453	658.118202	539.459350	1899.850792	

1	1
2	1
3	0
4	0

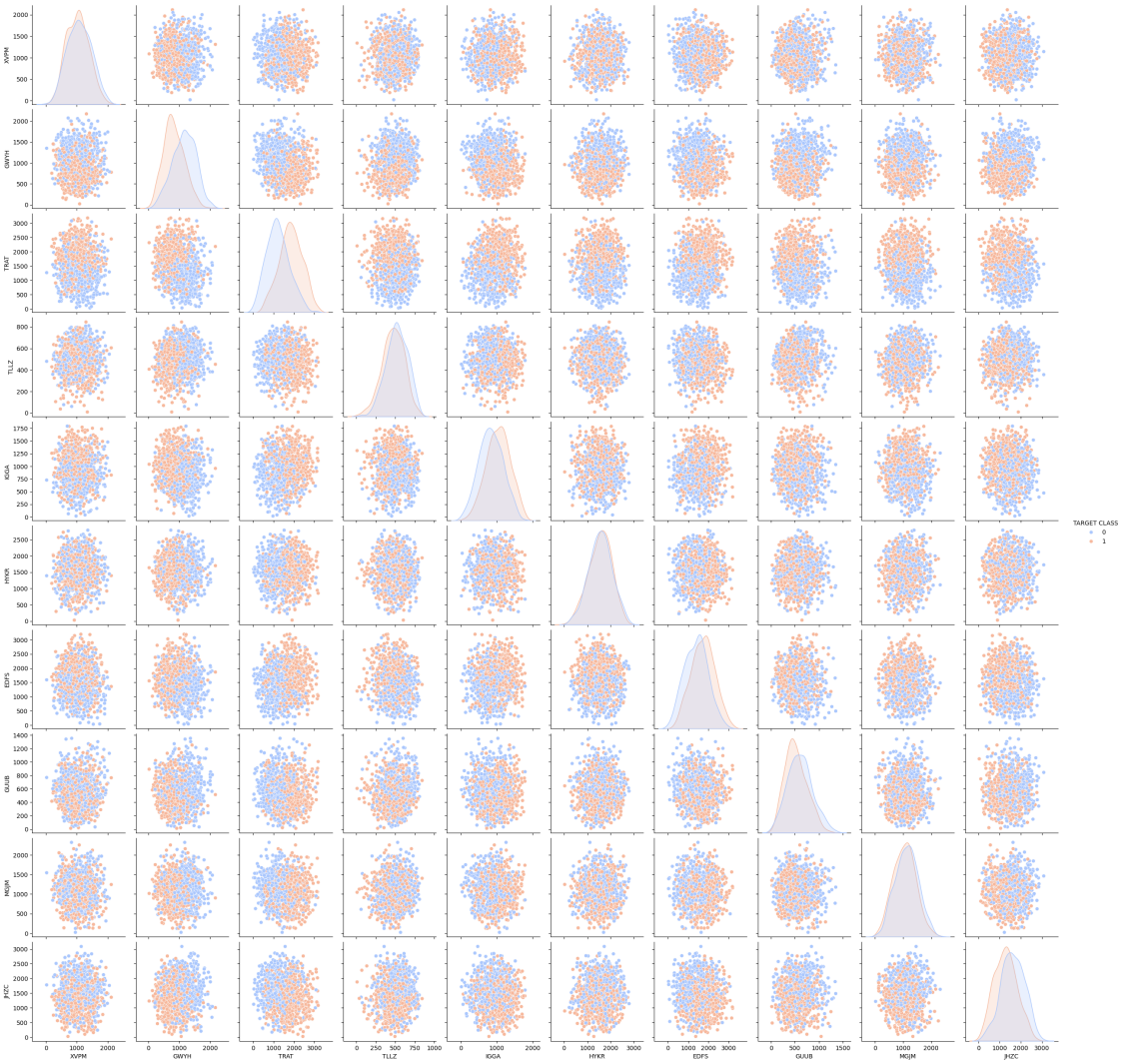
2 Análise exploratória de dados

Uma vez que esses dados são artificiais, vamos criar um grande pairplot com o Seaborn.

Use seaborn no DataFrame para criar um pairplot com a tonalidade indicada pela coluna TARGET CLASS.

[4]:
```
sns.pairplot(dados, hue='TARGET CLASS', palette='coolwarm')
```

[4]:
```
<seaborn.axisgrid.PairGrid at 0x1dd8ba93a10>
```



3 Padronize as variáveis

Hora de para padronizar as variáveis.

Import StandardScaler do Scikit-learn.

[5]:
```
from sklearn.preprocessing import StandardScaler
```

Crie um objeto StandardScaler() chamado scaler.

[6]:
```
scaler = StandardScaler()
```

Use o método fit() do objeto para treinar o modelo.

[7]:
```
scaler.fit(dados.drop('TARGET CLASS', axis=1))
```

[7]:
```
StandardScaler()
```

Use o método .transform () para transformar os parâmetros em uma versão padronizada.

[8]:
```
dados_padronizados = scaler.transform(dados.drop('TARGET CLASS', axis=1))
```

Converta os parâmetros padronizados em um DataFrame e verifique o cabeçalho desse DataFrame para garantir que a transform() funcionou.

[9]:
```
dados_padronizados = pd.DataFrame(data=dados_padronizados, columns=dados.
  ↪columns[:-1])
dados_padronizados.head()
```

[9]:

	XVPM	GWYH	TRAT	TLLZ	IGGA	HYKR	EDFS	GUUB	MGJM	JHZC
0	1.568522	-0.443435	1.619808	-0.958255	-1.128481	0.138336	0.980493	-0.932794	1.008313	-1.069627
1	-0.112376	-1.056574	1.741918	-1.504220	0.640009	1.081552	-1.182663	-0.461864	0.258321	-1.041546
2	0.660647	-0.436981	0.775793	0.213394	-0.053171	2.030872	-1.240707	1.149298	2.184784	0.342811
3	0.011533	0.191324	-1.433473	-0.100053	-1.507223	-1.753632	-1.183561	-0.888557	0.162310	-0.002793
4	-0.099059	0.820815	-0.904346	1.609015	-0.282065	-0.365099	-1.095644	0.391419	-1.365603	0.787762

4 Divisão treino-teste

Use o método train_test_split para dividir seus dados em um conjunto treino e teste.

[10]:
```
from sklearn.model_selection import train_test_split
```

```
X = dados_padronizados
y = dados['TARGET CLASS']

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

5 Usando o KNN

Importe o KNeighborClassifier do scikit learn.

[11]:
```
from sklearn.neighbors import KNeighborsClassifier
```

Crie uma instância do modelo KNN com n_neighbors = 1

[12]:
```
knn = KNeighborsClassifier(n_neighbors=1)
```

Ajuste este modelo KNN aos dados de treinamento.

[13]:
```
knn.fit(x_train, y_train)
```

[13]:
```
KNeighborsClassifier(n_neighbors=1)
```

6 Previsões e avaliações

Vamos avaliar o nosso modelo KNN!

Use o método de previsão para prever valores usando seu modelo KNN e X_test.

[14]:
```
y_pred = knn.predict(x_test)
```

** Crie uma matriz de confusão e um relatório de classificação. **

[18]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

[ ]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[110  36]
 [ 47 107]]
```

[16]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.75      0.73       146
           1       0.75      0.69      0.72       154

    accuracy                           0.72       300
   macro avg       0.72      0.72      0.72       300
weighted avg       0.73      0.72      0.72       300
```

7 Escolhendo o valor K

Vamos continuar usando o método do cotovelo para escolher um bom valor K!

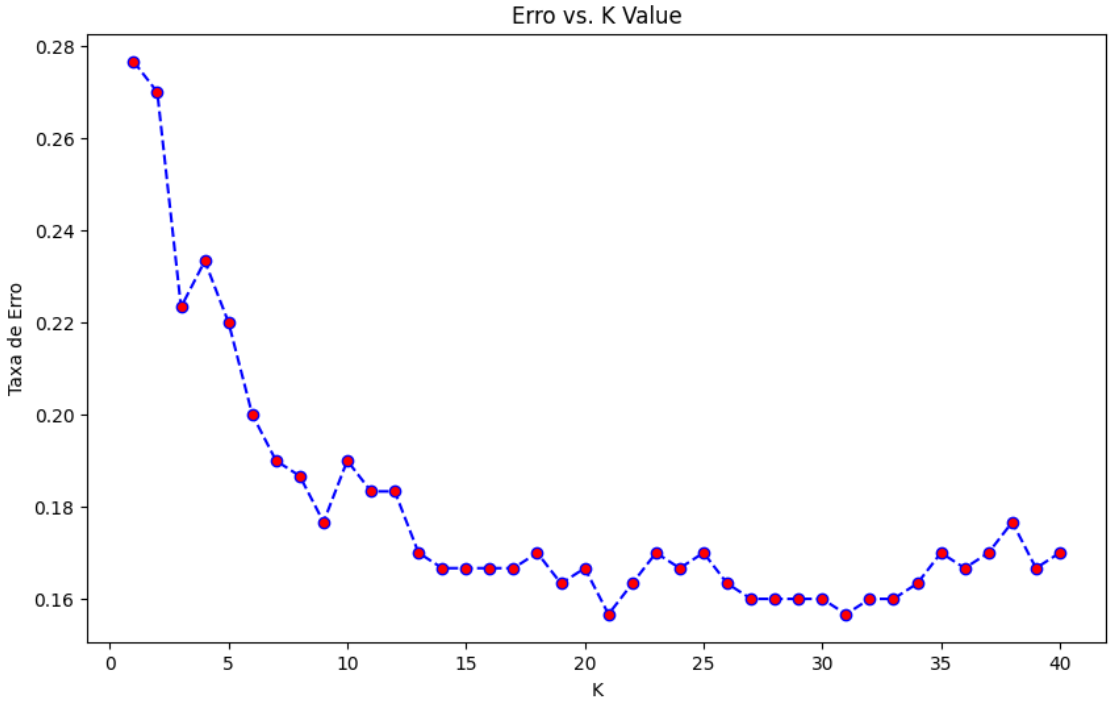
Crie um loop for que treine vários modelos KNN com valores k diferentes e, em seguida, mantenha um registro do error_rate para cada um desses modelos com uma lista. Consulte o notebook se você estiver confuso nesta etapa.

[19]:
```
error_rate = []

for k in range(1, 41):
    knn_temp = KNeighborsClassifier(n_neighbors=k)
    knn_temp.fit(x_train, y_train)
    pred_k = knn_temp.predict(x_test)
    error = np.mean(pred_k != y_test)
    error_rate.append(error)
```

Agora crie o seguinte gráfico usando as informações do seu loop.

[22]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1, 41), error_rate, color='blue', linestyle='dashed',␣
 ↪marker='o', markerfacecolor='red')
plt.title('Erro vs. K Value')
plt.xlabel('K')
plt.ylabel('Taxa de Erro')
plt.show()
```



7.1 Treine seu modelo novamente com novo valor K

** Treine novamente seu modelo com o melhor valor K (até você para decidir o que deseja) e re-faça o relatório de classificação e a matriz de confusão. **

[ ]:
```
melhor_k = error_rate.index(min(error_rate)) + 1
print("Melhor valor de K:", melhor_k)

knn_melhor = KNeighborsClassifier(n_neighbors=melhor_k)
knn_melhor.fit(x_train, y_train)


y_pred_melhor = knn_melhor.predict(x_test)


print(confusion_matrix(y_test, y_pred_melhor))
print(classification_report(y_test, y_pred_melhor))
```

```
Melhor valor de K: 21
[[125  21]
 [ 26 128]]
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       146
           1       0.86      0.83      0.84       154

    accuracy                           0.84       300
   macro avg       0.84      0.84      0.84       300
weighted avg       0.84      0.84      0.84       300
```