Proj3_resolvido

March 19, 2025

1 Projeto KNN

Neste projeto iremos trabalhar com o Iris dataset (iris.csv). Este é composto por 4 atributos (Comprimento e largura das sépalas e pétalas) de 150 amostras, sendo estas divididas em três espécies de iris (Setosa, Virginica e versicolor).

O objetivo é construir um modelo para classificar a espécie de iris.

1.1 Importar bibliotecas

** Importe algumas bibliotecas necessárias - numpy, matplotlib, seaborn e pandas**

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 Carregar o DataSet para um DataFrame Pandas

[2]:
```
df = pd.read_csv("iris.csv")
```

1.3 Análise exploratória de dados

Imprima os 5 primeiros registos do DataFrame

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

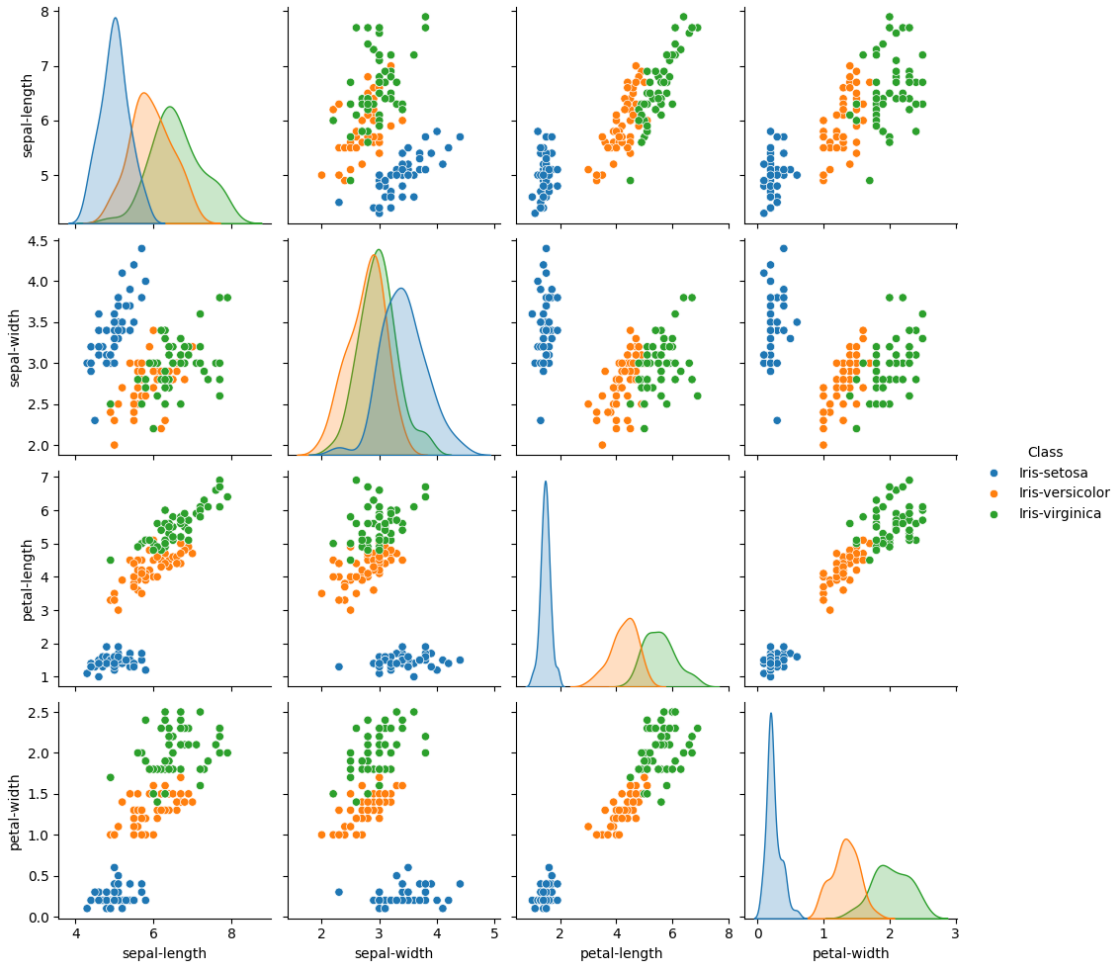
[3]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Visualize o relacionamento dos dados através de um pairplot.

[4]:
```
sns.pairplot(df, hue = 'Class')
```

[4]:
```
<seaborn.axisgrid.PairGrid at 0x147f3f970>
```



1.4 Codificar colunas de dados

Utilize o Label Encoder do sklearn para efetuar o mapeamento das colunas de dados categóricos em dados numéricos. Neste caso, apenas a coluna 'Class' contém dados categóricos.

No final, imprima a coluna 'Class' com os dados codificados.

[5]:
```
from sklearn import preprocessing

#creating label encoder
le = preprocessing.LabelEncoder()

#converting class
class_encoded=le.fit_transform(df['Class'])
print(class_encoded)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

1.5 Divisão dos dados

Divida os dados num conjunto de dados de treino e de teste.

O test_size deverá ser 0.3 e o random_state 101.

[6]:
```
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
# X_train, X_test, y_train, y_test =␣
 ↪train_test_split(list(zip(df['sepal-length'],df['sepal-width'],df['petal-length'],df['petal
 ↪class_encoded, test_size=0.3, random_state = 101) # 70% training and 30% test
X_train, X_test, y_train, y_test =␣
 ↪train_test_split(df[['sepal-length','sepal-width','petal-length','petal-width']],␣
 ↪class_encoded, test_size=0.3, random_state = 101) # 70% training and 30% test
```

1.6 Gerar o modelo KNN com K=2

[7]:
```
#Import knearest neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

#Create KNN Classifier
knn = KNeighborsClassifier(n_neighbors=2)

#Train the model using the training sets
knn.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = knn.predict(X_test)
```

2 Avaliação do modelo para K=2

Avalie o modelo gerado com K=2, calculando a 'Accuracy'

[8]:
```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9555555555555556
```

2.1 Escolher o melhor valor para K

Utilize o método Elbow (cotovelo) para escolher um bom Valor para K

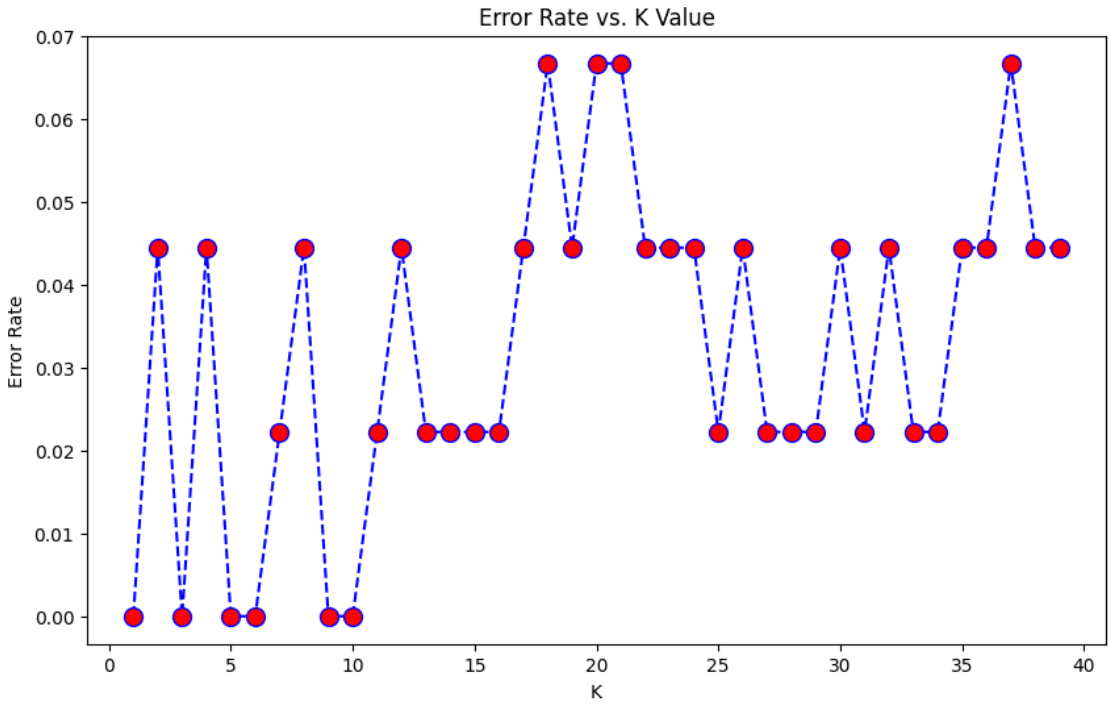
[9]:
```
from sklearn.metrics import classification_report, confusion_matrix

error_rate = []

for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test)) #média dos valores quando a␣
 ↪predição é diferente do y_test
```

[10]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,40), error_rate, color='blue', linestyle='dashed', marker='o',␣
 ↪markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

[10]:
```
Text(0, 0.5, 'Error Rate')
```



2.2 Treinar o modelo novamente, com o melhor valor para K

[11]:
```
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)
pred = knn.predict(X_test)
```

2.3 Avaliação do modelo com o melhor valor para K

Imprima a Confusion Matrix e o Classification Report

[12]:
```
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
[[13  0  0]
 [ 0 20  0]
 [ 0  0 12]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00        20
           2       1.00      1.00      1.00        12

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```