Librerías

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import datasets, metrics
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn import model_selection
```

Dataset

```
irisData = datasets.load_iris()

X = irisData.data
y = irisData.target
```

Divisón del dataset

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)

# Cross-Validation
kf = KFold(n_splits=5, random_state=7, shuffle=True)
```

Modelo

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier()
```

Evaluación

```
score = knn.score(X_train,y_train)
print("Metrica del modelo", score)

scores = cross_val_score(knn, X_train, y_train, cv=kf, scoring="accuracy")
# Accuracy
print("Metricas de Accuracy cross_validation", scores)
print("Media de Accuracy cross_validation", scores.mean())

preds = knn.predict(X_test)
score_pred = metrics.accuracy_score(y_test, preds)

print()
print("Metrica en Test", score_pred)

report = classification_report(y_test, preds)
print(report)
```

```
Metrica del modelo 0.9666666666666667
Metricas de Accuracy cross_validation [0.95833333 0.95833333 0.95833333 0.91666667 1.        ]
Media de Accuracy cross_validation 0.9583333333333333

Metrica en Test 1.0
              precision    recall  f1-score   support
```

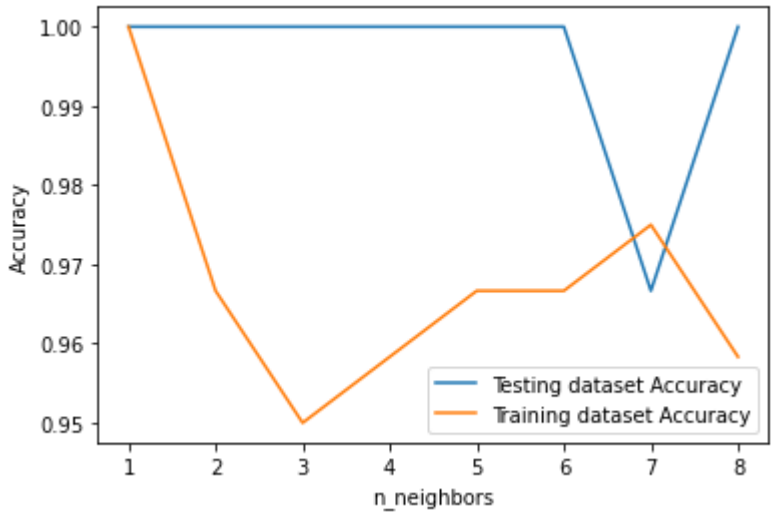
```
                0       1.00      1.00      1.00        10
                1       1.00      1.00      1.00         9
                2       1.00      1.00      1.00        11

         accuracy                           1.00        30
        macro avg       1.00      1.00      1.00        30
     weighted avg       1.00      1.00      1.00        30
```

```
1 y_pred = knn.predict(X_test)
```

```
1 from sklearn.metrics import accuracy_score
2 print("Accuracy", accuracy_score(y_test, y_pred)*100)
```

```
    Accuracy 100.0
```

```
 1 neighbors = np.arange(1, 9)
 2 train_accuracy = np.empty(len(neighbors))
 3 test_accuracy = np.empty(len(neighbors))
 4
 5 # Loop over K values
 6 for i, k in enumerate(neighbors):
 7     knn = KNeighborsClassifier(n_neighbors=k)
 8     knn.fit(X_train, y_train)
 9
10
11     train_accuracy[i] = knn.score(X_train, y_train)
12     test_accuracy[i] = knn.score(X_test, y_test)
13
14 plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
15 plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')
16
17 plt.legend()
18 plt.xlabel('n_neighbors')
19 plt.ylabel('Accuracy')
20 plt.show()
```




```
1 plt.figure(figsize = (15,5))
2 plt.subplot(1,2,1)
3 plt.scatter(X_test[:,0], X_test[:,1], c=y_pred, s=100,edgecolors='black')
4 plt.title("Predicted values", fontsize=20)
5 plt.show()
```