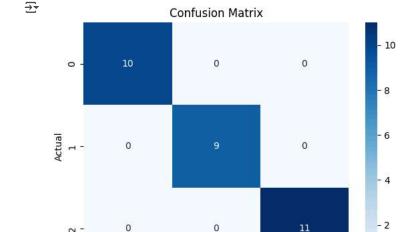
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model selection import train test split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.datasets import load_iris, load_diabetes
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import files
uploaded = files.upload()
     Choose Files iris.csv
       iris.csv(text/csv) - 3858 bytes, last modified: 4/7/2025 - 100% done
     Saving iris.csv to iris.csv
df = pd.read csv('iris.csv')
df
₹
           sepal_length sepal_width petal_length petal_width species
       0
                     5.1
                                   3.5
                                                               0.2
                                                                     setosa
                     4.9
                                   3.0
                                                 1.4
                                                               0.2
                                                                     setosa
       2
                     4.7
                                   3.2
                                                 1.3
                                                               0.2
                                                                     setosa
       3
                     4.6
                                   3.1
                                                 1.5
                                                               0.2
                                                                     setosa
       4
                     5.0
                                   3.6
                                                 1.4
                                                               0.2
                                                                     setosa
                                                                ...
      145
                     6.7
                                   3.0
                                                 5.2
                                                               2.3
                                                                    virginica
      146
                     6.3
                                   2.5
                                                 5.0
                                                               1.9
                                                                    virginica
                                                 52
      147
                     6.5
                                   3.0
                                                               2.0
                                                                    virginica
      148
                     6.2
                                                                    virginica
                                   3.4
                                                 5.4
                                                               2.3
      149
                     5.9
                                   3.0
                                                 5.1
                                                               1.8 virginica
     150 rows × 5 columns
 Next steps: ( Generate code with df )

    View recommended plots

                                                                   New interactive sheet
df['species'] = df['species'].astype('category').cat.codes
X = df.drop('species', axis=1)
y = df['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
param grid = {'n neighbors': list(range(1, 21))}
knn = KNeighborsClassifier()
grid = GridSearchCV(knn, param_grid, cv=5)
grid.fit(X_train, y_train)
∓
                     GridSearchCV
                   best_estimator_:
                KNeighborsClassifier
             ▶ KNeighborsClassifier
```

```
best_k = grid.best_params_['n_neighbors']
print(f"Best k value: {best_k}")
→ Best k value: 3
knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(X_train, y_train)
<del>_</del>
           KNeighborsClassifier
     KNeighborsClassifier(n_neighbors=3)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy Score: {accuracy:.2f}")
Accuracy Score: 1.00
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
→ Confusion Matrix:
     [[10 0 0]
      [090]
      [0011]]
print("Classification Report:")
print(classification_report(y_test, y_pred))
→ Classification Report:
                                recall f1-score
                   precision
                                                   support
                0
                        1.00
                                  1.00
                                            1.00
                                                        10
                        1.00
                                            1.00
                                                         9
                1
                                  1.00
                2
                        1.00
                                  1.00
                                            1.00
                                                        11
                                            1.00
                                                        30
         accuracy
                        1.00
        macro avg
                                  1.00
                                            1.00
                                                        30
     weighted avg
                        1.00
                                  1.00
                                            1.00
                                                        30
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



1 Predicted

0

2

- 0