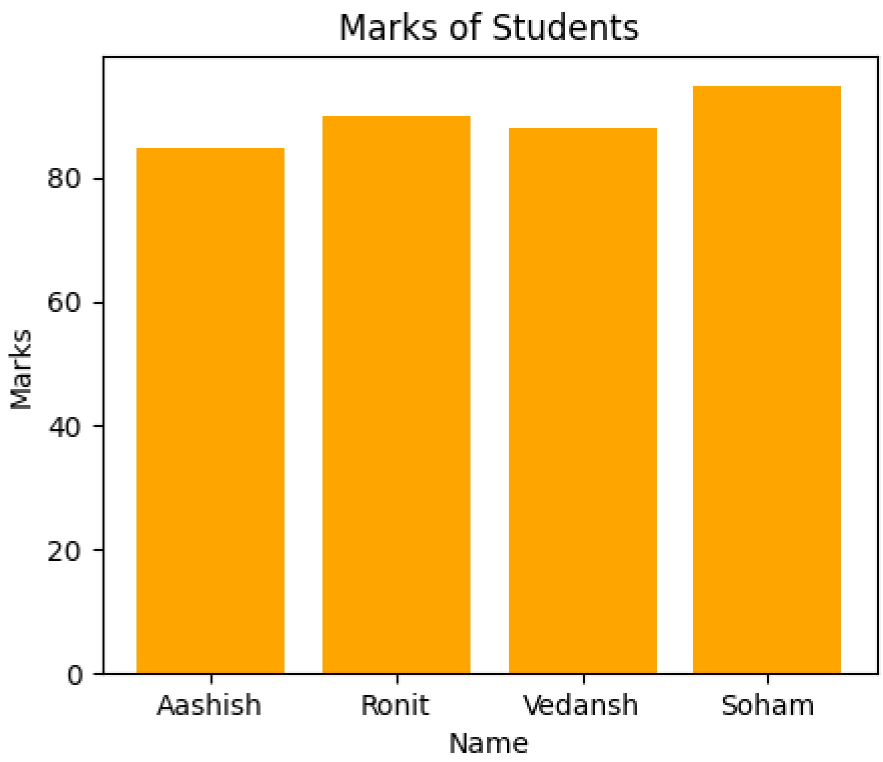
Code :-

```python
# Assignment 4: Demonstrate Python Libraries
import math, numpy as np, pandas as pd

from scipy import stats

import matplotlib.pyplot as plt


# 1️⃣ Math Library

print("MATH LIBRARY:")

print("Square root of 25:", math.sqrt(25))

print("Factorial of 5:", math.factorial(5))

print("Ceil(3.2):", math.ceil(3.2), " Floor(3.8):", math.floor(3.8),
"\n")


# 2️⃣ NumPy & SciPy

arr = np.random.randint(1, 100, 10)

print("NUMPY ARRAY:", arr)

print("Mean:", np.mean(arr), " Median:", np.median(arr))

print("Std Dev:", np.std(arr), " Var:", np.var(arr))

print("Mode using SciPy:", stats.mode(arr, keepdims=True).mode[0],
"\n")


# 3️⃣ Pandas

data = {"Name": ["Aashish","Ronit","Vedansh","Soham"],

        "Age": [20, 21, 22, 23],

        "Marks": [85, 90, 88, 95]}

df = pd.DataFrame(data)

print("PANDAS DATAFRAME:\n", df)

print("\nSummary:\n", df.describe(), "\n")


# 4️⃣ Matplotlib Visualization
```

```
plt.figure(figsize=(5,4))

plt.bar(df["Name"], df["Marks"], color='orange')

plt.title("Marks of Students")

plt.xlabel("Name")

plt.ylabel("Marks")

plt.show()
```

Output:-



Code:-

```
# assignment5_stats.py
# Basic arithmetic and statistical calculations (Mean, Median, Std Dev)

import numpy as np

# Arithmetic Operations
a = 15
b = 4
print("Arithmetic Operations:")
print(f"Addition: {a} + {b} = {a + b}")
print(f"Subtraction: {a} - {b} = {a - b}")
print(f"Multiplication: {a} * {b} = {a * b}")
print(f"Division: {a} / {b} = {a / b}")
print(f"Modulus: {a} % {b} = {a % b}")
print(f"Exponentiation: {a} ** {b} = {a ** b}")
print()

# Statistical Calculations
data = [10, 20, 30, 40, 50]
mean = np.mean(data)
median = np.median(data)
std_dev = np.std(data)

print("Statistical Calculations:")
print(f"Data: {data}")
print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Standard Deviation: {std_dev}")
```

Output

```
(.venv) PS C:\Users\Lenovo\titanic_project> pip install numpy
>>
Requirement already satisfied: numpy in c:\users\lenovo\titanic_project\.venv\lib\site-p
ackages (2.3.4)
(.venv) PS C:\Users\Lenovo\titanic_project> python assignment5_stats.py
>>
Arithmetic Operations:
Addition: 15 + 4 = 19
Subtraction: 15 - 4 = 11
Multiplication: 15 * 4 = 60
Division: 15 / 4 = 3.75
Modulus: 15 % 4 = 3
Exponentiation: 15 ** 4 = 50625

Statistical Calculations:
Data: [10, 20, 30, 40, 50]
Mean: 30.0
Median: 30.0
Standard Deviation: 14.142135623730951
(.venv) PS C:\Users\Lenovo\titanic_project>
```

Code :-

```
# Assignment 6: Data Preprocessing - Missing Values, Encoding,
Normalization
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, MinMaxScaler,
StandardScaler

# Step 1: Load Dataset (Titanic dataset from CSV or create sample)
try:
    df = pd.read_csv("titanic.csv")   # if you have a Titanic CSV
    print("Loaded titanic.csv successfully!\n")
except:
    print("No titanic.csv found, using sample dataset...\n")
    df = pd.DataFrame({
        "Name": ["Aashish", "Saloni", "Vedansh", "Poornima", "Aditya"],
        "Age": [20, np.nan, 22, 19, 23],
        "Gender": ["Male", "Female", "Male", "Female", "Male"],
        "Salary": [25000, 27000, np.nan, 30000, 24000]
    })

# Step 2: Explore Data
print("Original Data:\n", df, "\n")
print("Missing values per column:\n", df.isnull().sum(), "\n")

# Step 3: Handle Missing Values
df["Age"].fillna(df["Age"].mean(), inplace=True)
df["Salary"].fillna(df["Salary"].mean(), inplace=True)
print("After handling missing values:\n", df, "\n")

# Step 4: Encode Categorical Data
le = LabelEncoder()
df["Gender_Encoded"] = le.fit_transform(df["Gender"])  # Male=1,
Female=0
print("After Encoding Categorical Data:\n", df, "\n")

# Step 5: Normalization
scaler = MinMaxScaler()
df[["Age_Norm", "Salary_Norm"]] = scaler.fit_transform(df[["Age",
"Salary"]])
print("After Normalization (Min-Max Scaling):\n", df, "\n")

# Step 6: Standardization (optional)
```

```python
std_scaler = StandardScaler()
df[["Age_Std", "Salary_Std"]] = std_scaler.fit_transform(df[["Age",
"Salary"]])
print("After Standardization (Z-score):\n", df, "\n")

# Final clean data
print("=== Final Preprocessed Data ===")
print(df)
```

Outpu :-

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
PS C:\Users\Lenovo\titanic_project> & C:/Users/Lenovo/titanic_project/.venv/Scripts/Activate.p
s1
(.venv) PS C:\Users\Lenovo\titanic_project> & C:/Users/Lenovo/titanic_project/.venv/Scripts/py
thon.exe c:/Users/Lenovo/titanic_project/assignment6_preprocessing.py
No titanic.csv found, using sample dataset...

Original Data:
       Name   Age  Gender   Salary
0   Aashish  20.0    Male  25000.0
1    Saloni   NaN  Female  27000.0
2   Vedansh  22.0    Male      NaN
3  Poornima  19.0  Female  30000.0
4    Aditya  23.0    Male  24000.0

Missing values per column:
 Name      0
Age       1
Gender    0
Salary    1
dtype: int64

c:\Users\Lenovo\titanic_project\assignment6_preprocessing.py:24: FutureWarning: A value is try
ing to be set on a copy of a DataFrame or Series through chained assignment using an inplace m
ethod.
The behavior will change in pandas 3.0. This inplace method will never work because the interm
ediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu
e}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplac
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL Python

```
(.venv) PS C:\Users\Lenovo\titanic_project> & C:/Users/Lenovo/titanic_project/.venv/Scripts/py
thon.exe c:/Users/Lenovo/titanic_project/assignment6_preprocessing.py

  df["Age"].fillna(df["Age"].mean(), inplace=True)
c:\Users\Lenovo\titanic_project\assignment6_preprocessing.py:25: FutureWarning: A value is try
ing to be set on a copy of a DataFrame or Series through chained assignment using an inplace m
ethod.
The behavior will change in pandas 3.0. This inplace method will never work because the interm
ediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: valu
e}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplac
e on the original object.
```

```
  df["Salary"].fillna(df["Salary"].mean(), inplace=True)
After handling missing values:
       Name   Age  Gender   Salary
0   Aashish  20.0    Male  25000.0
1    Saloni  21.0  Female  27000.0
2   Vedansh  22.0    Male  26500.0
3  Poornima  19.0  Female  30000.0
4    Aditya  23.0    Male  24000.0

After Encoding Categorical Data:
       Name   Age  Gender   Salary  Gender_Encoded
0   Aashish  20.0    Male  25000.0               1
1    Saloni  21.0  Female  27000.0               0
2   Vedansh  22.0    Male  26500.0               1
3  Poornima  19.0  Female  30000.0               0
4    Aditya  23.0    Male  24000.0               1

After Normalization (Min-Max Scaling):
       Name   Age  Gender   Salary  Gender_Encoded  Age_Norm  Salary_Norm
0   Aashish  20.0    Male  25000.0               1      0.25     0.166667
1    Saloni  21.0  Female  27000.0               0      0.50     0.500000
2   Vedansh  22.0    Male  26500.0               1      0.75     0.416667
3  Poornima  19.0  Female  30000.0               0      0.00     1.000000
4    Aditya  23.0    Male  24000.0               1      1.00     0.000000
```

```
After Standardization (Z-score):
       Name   Age  Gender   Salary  ...  Age_Norm  Salary_Norm   Age_Std  Salary_Std
0   Aashish  20.0    Male  25000.0  ...      0.25     0.166667 -0.707107   -0.731925
1    Saloni  21.0  Female  27000.0  ...      0.50     0.500000  0.000000    0.243975
2   Vedansh  22.0    Male  26500.0  ...      0.75     0.416667  0.707107    0.000000
3  Poornima  19.0  Female  30000.0  ...      0.00     1.000000 -1.414214    1.707825
4    Aditya  23.0    Male  24000.0  ...      1.00     0.000000  1.414214   -1.219875

[5 rows x 9 columns]
4    Aditya  23.0    Male  24000.0               1

After Normalization (Min-Max Scaling):
       Name   Age  Gender   Salary  Gender_Encoded  Age_Norm  Salary_Norm
0   Aashish  20.0    Male  25000.0               1      0.25     0.166667
1    Saloni  21.0  Female  27000.0               0      0.50     0.500000
2   Vedansh  22.0    Male  26500.0               1      0.75     0.416667
3  Poornima  19.0  Female  30000.0               0      0.00     1.000000
4    Aditya  23.0    Male  24000.0               1      1.00     0.000000

After Standardization (Z-score):
       Name   Age  Gender   Salary  ...  Age_Norm  Salary_Norm   Age_Std  Salary_Std
0   Aashish  20.0    Male  25000.0  ...      0.25     0.166667 -0.707107   -0.731925
1    Saloni  21.0  Female  27000.0  ...      0.50     0.500000  0.000000    0.243975
2   Vedansh  22.0    Male  26500.0  ...      0.75     0.416667  0.707107    0.000000
3  Poornima  19.0  Female  30000.0  ...      0.00     1.000000 -1.414214    1.707825
```

```
=== Final Preprocessed Data ===
       Name   Age  Gender   Salary  ...  Age_Norm  Salary_Norm   Age_Std  Salary_Std

=== Final Preprocessed Data ===
       Name   Age  Gender   Salary  ...  Age_Norm  Salary_Norm   Age_Std  Salary_Std
0   Aashish  20.0    Male  25000.0  ...      0.25     0.166667 -0.707107   -0.731925
1    Saloni  21.0  Female  27000.0  ...      0.50     0.500000  0.000000    0.243975
       Name   Age  Gender   Salary  ...  Age_Norm  Salary_Norm   Age_Std  Salary_Std
0   Aashish  20.0    Male  25000.0  ...      0.25     0.166667 -0.707107   -0.731925
1    Saloni  21.0  Female  27000.0  ...      0.50     0.500000  0.000000    0.243975
0   Aashish  20.0    Male  25000.0  ...      0.25     0.166667 -0.707107   -0.731925
1    Saloni  21.0  Female  27000.0  ...      0.50     0.500000  0.000000    0.243975
1    Saloni  21.0  Female  27000.0  ...      0.50     0.500000  0.000000    0.243975
2   Vedansh  22.0    Male  26500.0  ...      0.75     0.416667  0.707107    0.000000
3  Poornima  19.0  Female  30000.0  ...      0.00     1.000000 -1.414214    1.707825
4    Aditya  23.0    Male  24000.0  ...      1.00     0.000000  1.414214   -1.219875

[5 rows x 9 columns]
(.venv) PS C:\Users\Lenovo\titanic_project>
```

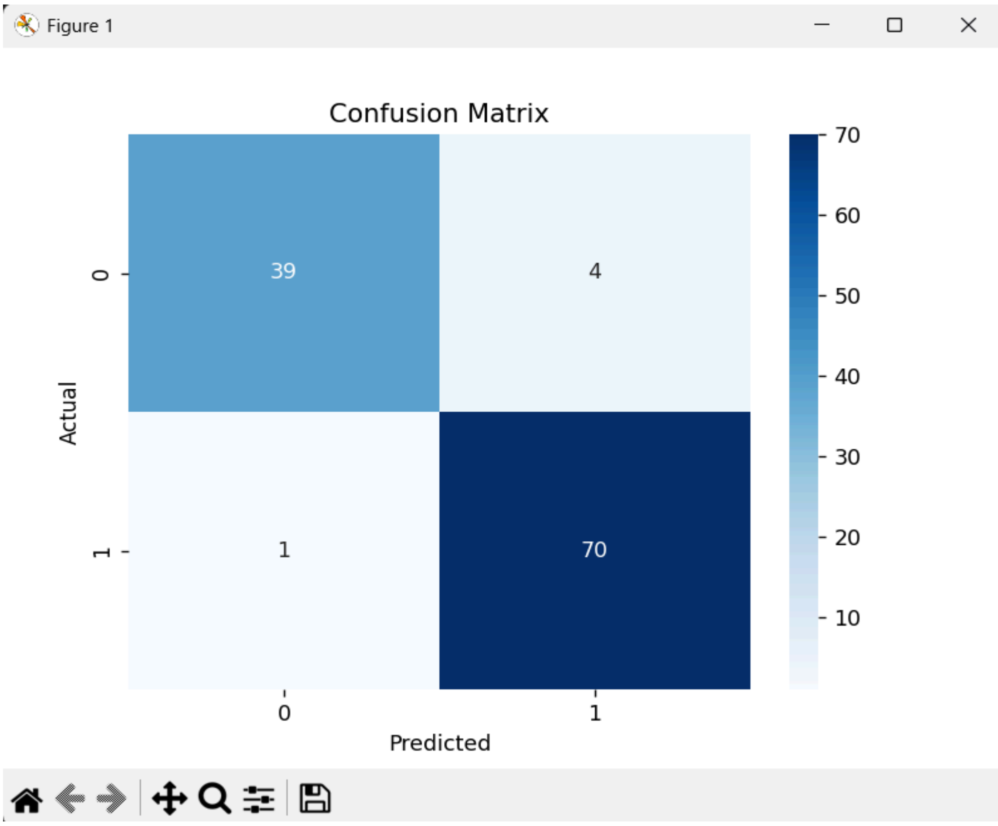
Code:

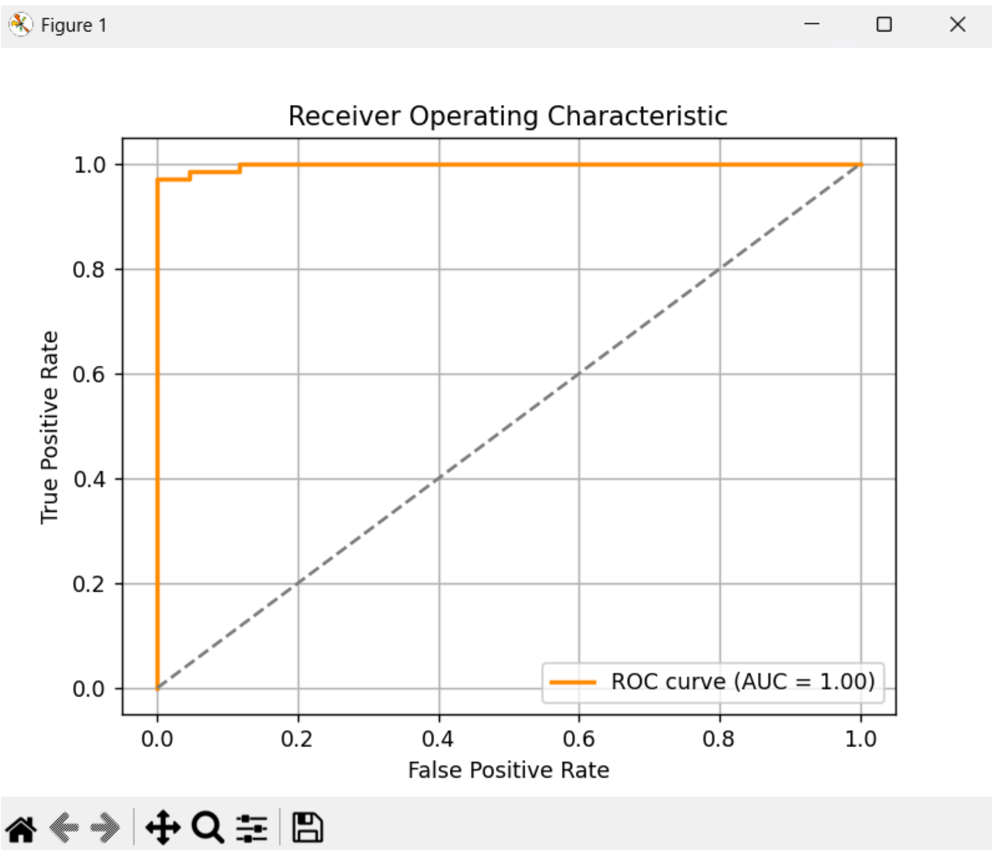
```
# Step 1: Import libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns

# Step 2: Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Step 3: Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Train Logistic Regression model
model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)

# Step 5: Predictions
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]  # Probabilities for ROC curve

# Step 6: Evaluation
```

```python
# Accuracy
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy: {acc:.4f}")

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

Output:

Accuracy: 0.9561





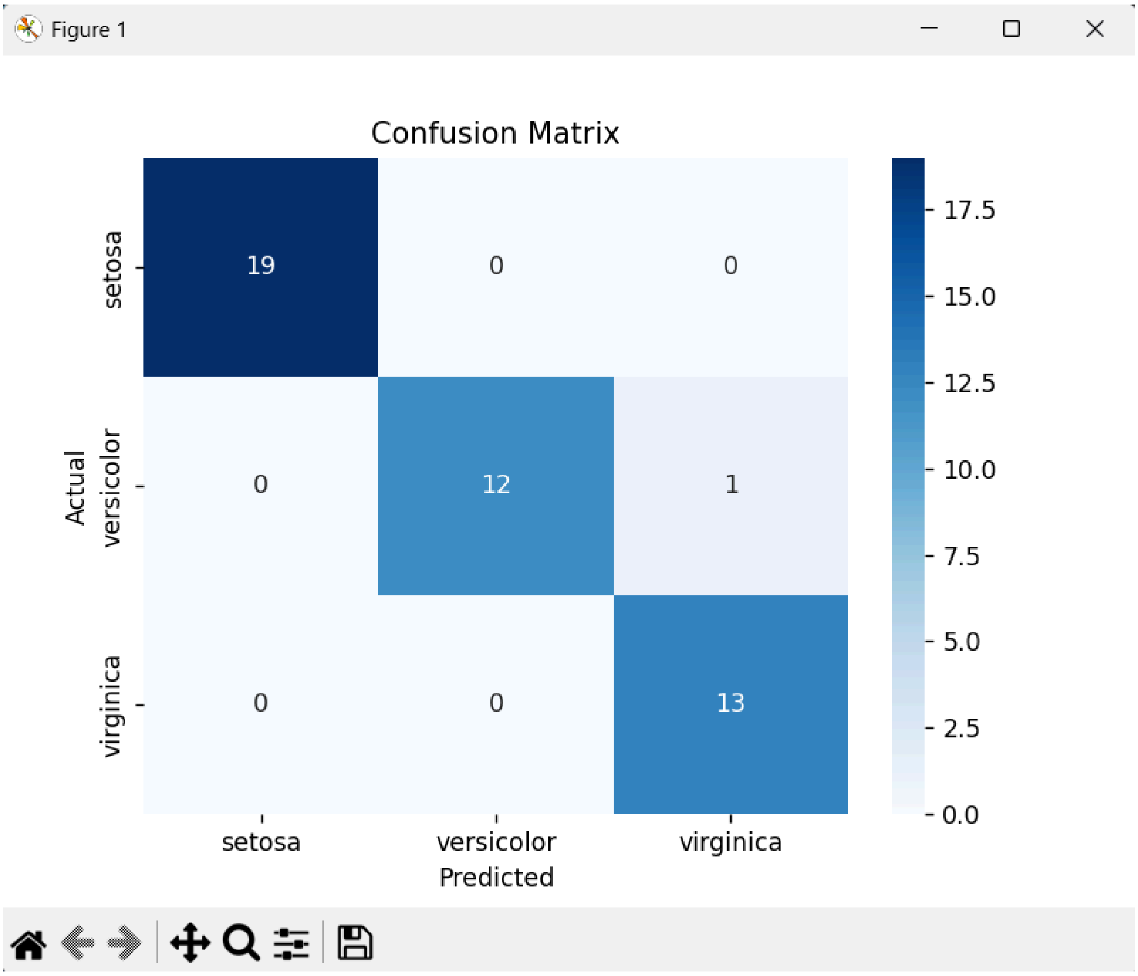
Code:

```
# Step 1: Import libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Step 2: Load dataset
iris = load_iris()
X = iris.data
y = iris.target

# Step 3: Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 4: Train Naive Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# Step 5: Make predictions
y_pred = model.predict(X_test)

# Step 6: Evaluate the model


# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}"
```

```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap='Blues', xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()


# Precision, Recall, F1-score
print("\n Classification Report:\n")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

Output:

Accuracy: 0.9778



Classification Report:

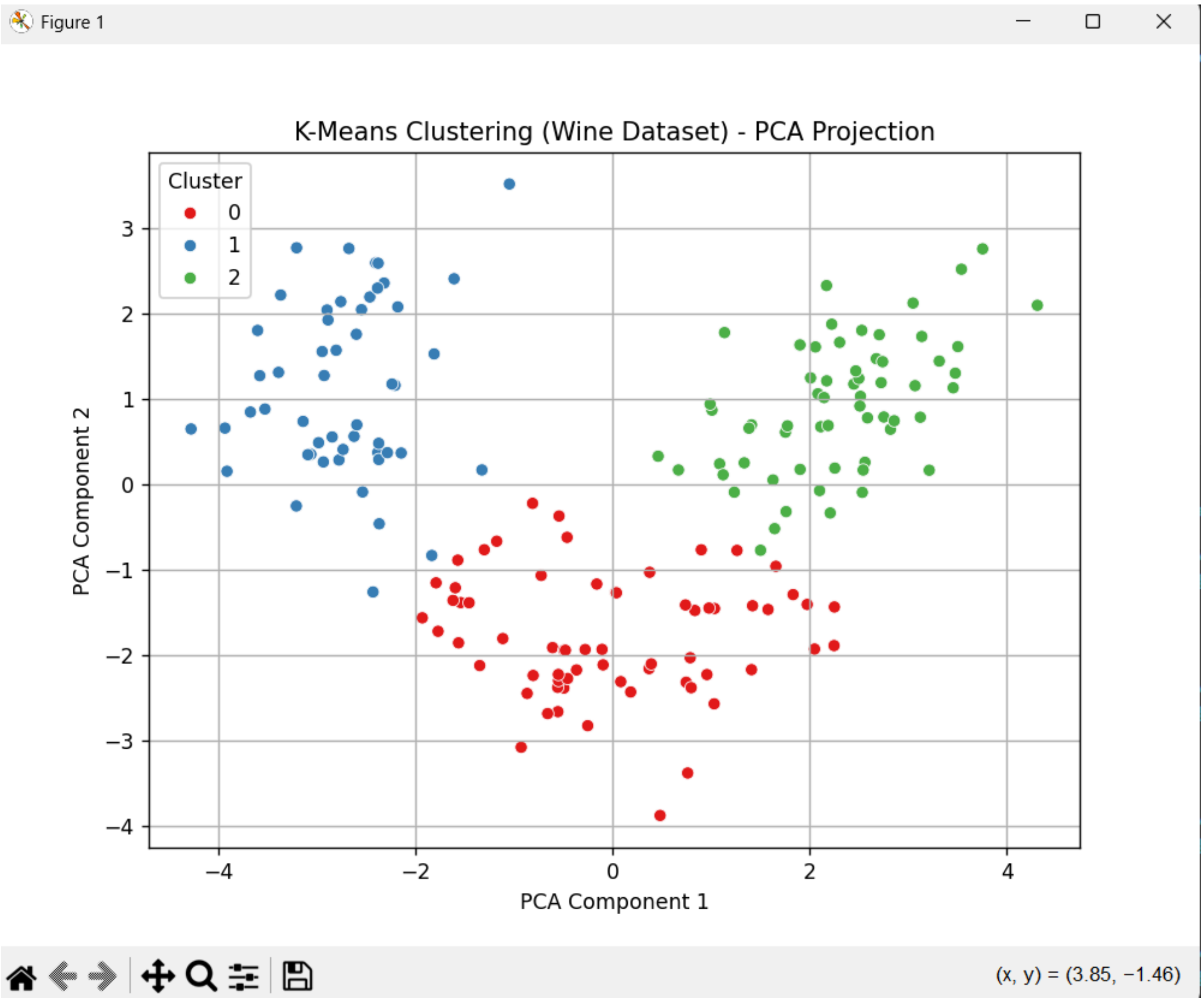
	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	19
versicolor	1.00	0.92	0.96	13
virginica	0.93	1.00	0.96	13
accuracy			0.98	45
macro avg	0.98	0.97	0.97	45
weighted avg	0.98	0.98	0.98	45

Code:

```
# Step 1: Import required libraries
from sklearn.datasets import load_wine
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns

# Step 2: Load the dataset
data = load_wine()
X = data.data
feature_names = data.feature_names

# Step 3: Normalize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 4: Apply K-Means clustering
k = 3  # since wine dataset has 3 classes
kmeans = KMeans(n_clusters=k, random_state=42)
labels = kmeans.fit_predict(X_scaled)

# Step 5: Evaluate with Silhouette Score
sil_score = silhouette_score(X_scaled, labels)
print(f"Silhouette Score: {sil_score:.4f}")
```

```
# Step 6: Visualize clusters using PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=labels, palette='Set1')
plt.title("K-Means Clustering (Wine Dataset) - PCA Projection")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.grid(True)
plt.legend(title='Cluster')
plt.show()
```

Output:



Silhouette Score: 0.2849

Code:

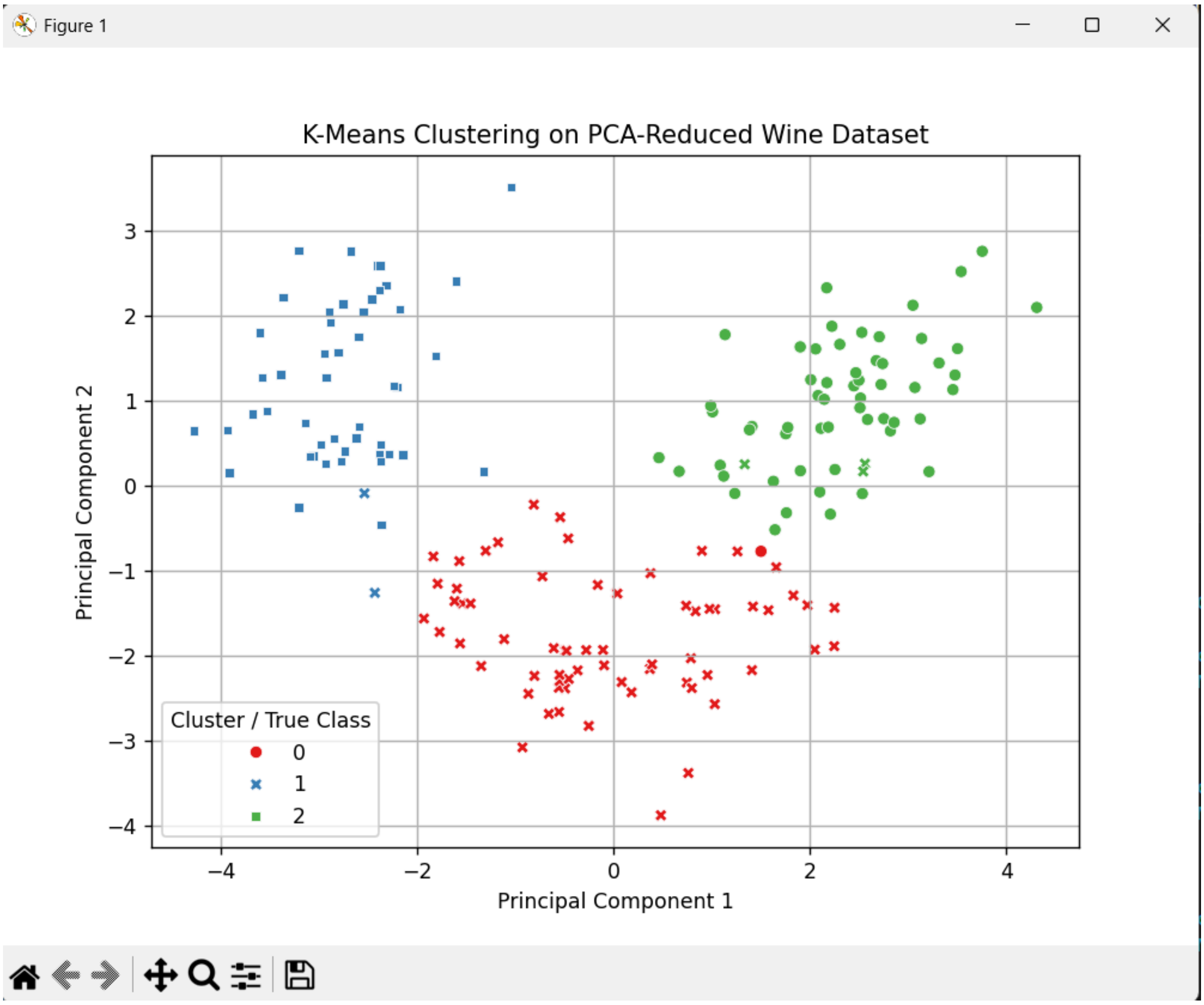
```
# Step 1: Import libraries
from sklearn.datasets import load_wine
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns

# Step 2: Load Wine dataset
data = load_wine()
X = data.data
y = data.target  # true labels (for comparison only)

# Step 3: Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 4: Apply PCA to reduce to 2 dimensions
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

print("Explained Variance Ratio:", pca.explained_variance_ratio_)
print("Total Variance Retained:", sum(pca.explained_variance_ratio_))

# Step 5: Apply K-Means on 2D PCA data
kmeans = KMeans(n_clusters=3, random_state=42)
```

```
cluster_labels = kmeans.fit_predict(X_pca)

# Step 6: Visualize Clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=cluster_labels, palette="Set1", style=y)
plt.title("K-Means Clustering on PCA-Reduced Wine Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.grid(True)
plt.legend(title='Cluster / True Class')
plt.show()

# Step 7: Evaluate cluster quality
score = silhouette_score(X_pca, cluster_labels)
print(f"Silhouette Score: {score:.4f}")
```

Output:

Explained Variance Ratio: [0.36198848 0.1920749]

Total Variance Retained: 0.5540633835693526



Silhouette Score: 0.5602