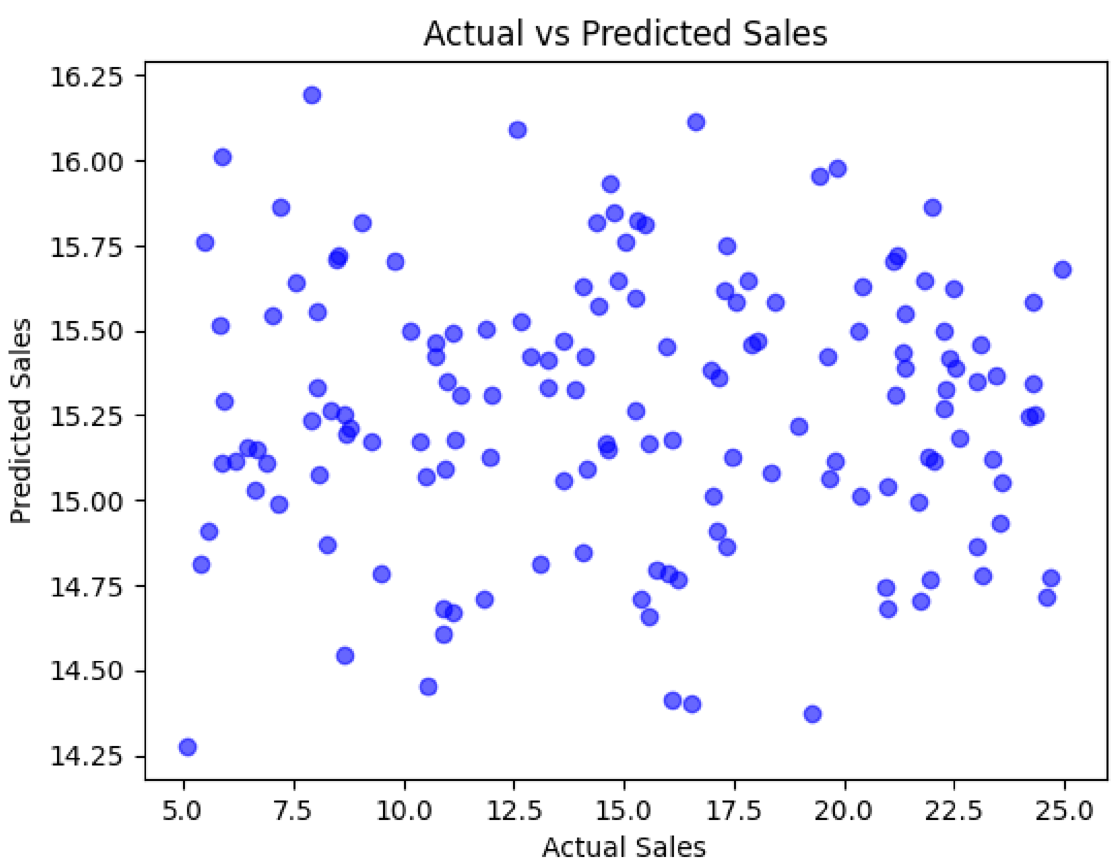
```
Question 1)

# 1. Import Libraries
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# 2. Creating the 'sales' Dataset
np.random.seed(0)  # For reproducibility
data = pd.DataFrame({
    'ID': range(1, 501),
    'TV': np.random.uniform(50, 300, 500),
    'Radio': np.random.uniform(10, 100, 500),
    'Newspaper': np.random.uniform(5, 50, 500),
    'Sales': np.random.uniform(5, 25, 500)
})
print(data.head())  # Previewing the first 5 rows

# 3. Splitting the Data into Independent and Target Variables
X = data[['TV', 'Radio', 'Newspaper']].values  # Independent variables
y = data['Sales'].values  # Target variable

# Splitting into Training and Testing Sets (7:3 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0)

print("Training Set Shape:", X_train.shape, y_train.shape)
print("Testing Set Shape:", X_test.shape, y_test.shape)

# 4. Building a Linear Regression Model
linear_regressor = LinearRegression()  # Creating the linear
regression model
linear_regressor.fit(X_train, y_train)  # Fitting the model

# Making predictions on the test set
y_pred = linear_regressor.predict(X_test)

# 5. Evaluating the Model
print("Model Coefficients:", linear_regressor.coef_)
print("Model Intercept:", linear_regressor.intercept_)

# Calculating R-squared and Mean Squared Error
from sklearn.metrics import r2_score, mean_squared_error
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print(f"R-squared: {r2:.2f}")
```

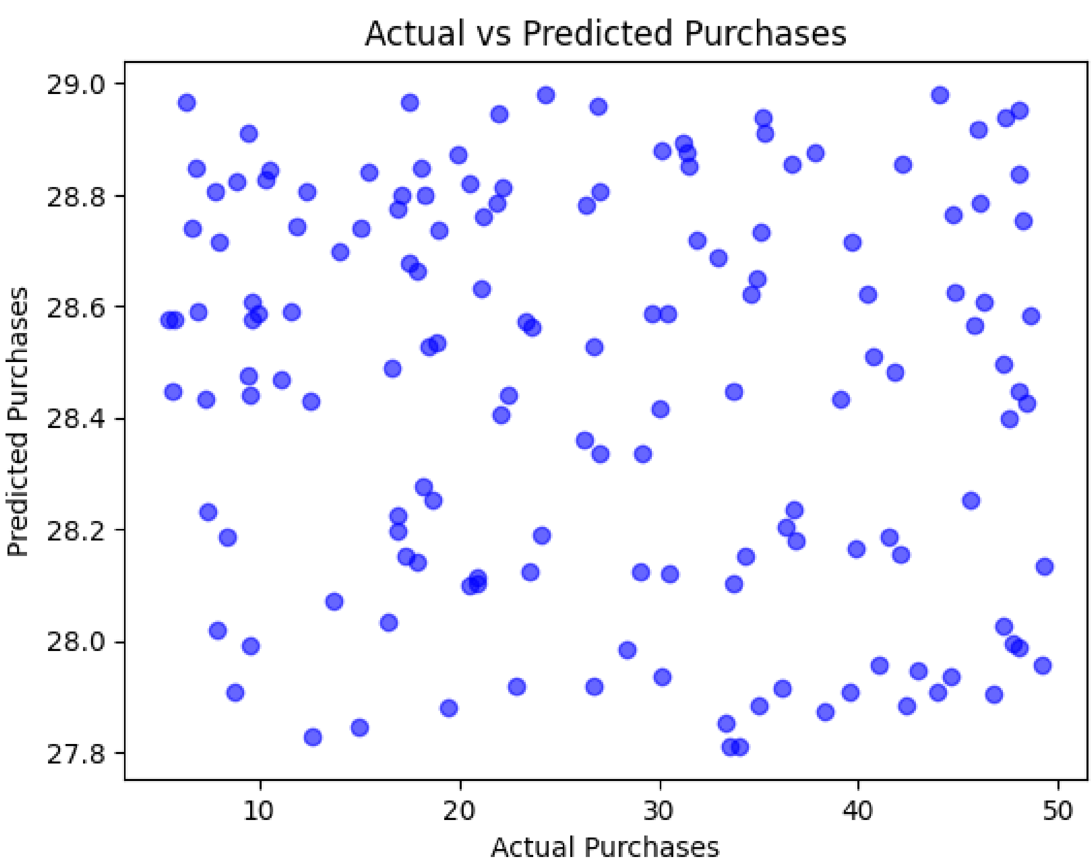
```
print(f"Mean Squared Error: {mse:.2f}")

# Plotting Actual vs Predicted Sales
plt.scatter(y_test, y_pred, color='blue', alpha=0.6)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs Predicted Sales')
plt.show()

   ID          TV      Radio  Newspaper      Sales
0   1  187.203376  37.934274  31.679612  13.935866
1   2  228.797342  43.573138   5.452866  21.739807
2   3  200.690844  57.247340  26.412179   9.436481
3   4  186.220796  77.553552  36.894668  14.878905
4   5  155.913700  40.015672   6.978894  23.592375
Training Set Shape: (350, 3) (350,)
Testing Set Shape: (150, 3) (150,)
Model Coefficients: [-0.00400006 -0.00810181  0.01809161]
Model Intercept: 15.87427913070574
R-squared: -0.01
Mean Squared Error: 32.80
```



```
Question 2)

# 1. Import Libraries
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# 2. Creating the 'realestate' Dataset
np.random.seed(0)  # For reproducibility
realestate_data = pd.DataFrame({
    'ID': range(1, 501),
    'Flat': np.random.uniform(50, 500, 500),
    'Houses': np.random.uniform(20, 300, 500),
    'Purchases': np.random.uniform(5, 50, 500)
})
print(realestate_data.head())  # Previewing the first 5 rows

# 3. Splitting the Data into Independent and Target Variables
X = realestate_data[['Flat', 'Houses']].values  # Independent
variables
y = realestate_data['Purchases'].values  # Target variable

# Splitting into Training and Testing Sets (7:3 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0)

print("Training Set Shape:", X_train.shape, y_train.shape)
print("Testing Set Shape:", X_test.shape, y_test.shape)

# 4. Building a Linear Regression Model
linear_regressor = LinearRegression()  # Creating the linear
regression model
linear_regressor.fit(X_train, y_train)  # Fitting the model

# Making predictions on the test set
y_pred = linear_regressor.predict(X_test)

# 5. Evaluating the Model
print("Model Coefficients:", linear_regressor.coef_)
print("Model Intercept:", linear_regressor.intercept_)

# Calculating R-squared and Mean Squared Error
from sklearn.metrics import r2_score, mean_squared_error
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print(f"R-squared: {r2:.2f}")
```

```
print(f"Mean Squared Error: {mse:.2f}")

# Plotting Actual vs Predicted Purchases
plt.scatter(y_test, y_pred, color='blue', alpha=0.6)
plt.xlabel('Actual Purchases')
plt.ylabel('Predicted Purchases')
plt.title('Actual vs Predicted Purchases')
plt.show()
```

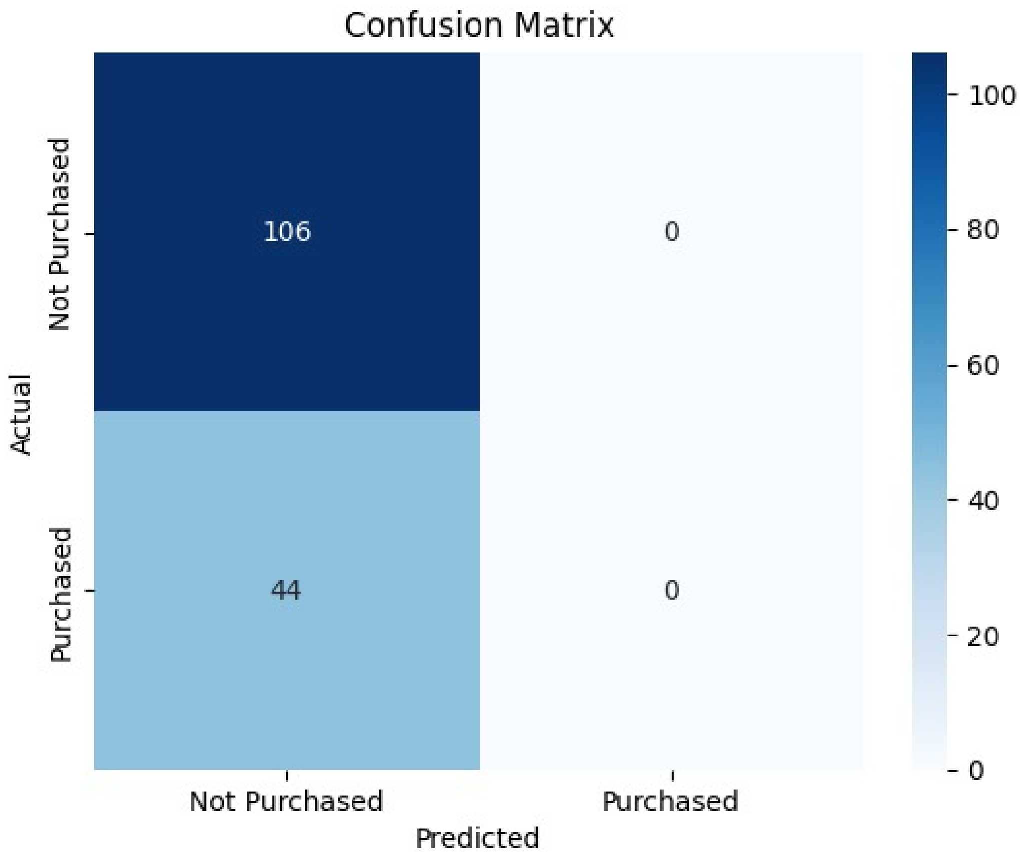
```
   ID        Flat      Houses  Purchases
0   1  296.966077  106.906631  31.679612
1   2  371.835215  124.449762   5.452866
2   3  321.243519  166.991724  26.412179
3   4  295.197432  230.166606  36.894668
4   5  240.644660  113.382090   6.978894
Training Set Shape: (350, 2) (350,)
Testing Set Shape: (150, 2) (150,)
Model Coefficients: [ 0.00023652 -0.0041004 ]
Model Intercept: 28.9991454239178
R-squared: -0.02
Mean Squared Error: 181.83
```



```
Question 3)

# 1. Import Libraries
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

# 2. Creating the 'User' Dataset
np.random.seed(0)  # For reproducibility
user_data = pd.DataFrame({
    'User ID': range(1, 501),
    'Gender': np.random.choice(['Male', 'Female'], size=500),
    'Age': np.random.randint(18, 60, size=500),
    'EstimatedSalary': np.random.uniform(20000, 150000, 500),
    'Purchased': np.random.choice([0, 1], size=500, p=[0.7, 0.3])
})
print(user_data.head())  # Previewing the first 5 rows

# Encoding Gender Column (Male: 1, Female: 0)
user_data['Gender'] = user_data['Gender'].map({'Male': 1, 'Female':
0})

# 3. Splitting the Data into Independent and Target Variables
X = user_data[['Gender', 'Age', 'EstimatedSalary']].values  #
Independent variables
y = user_data['Purchased'].values  # Target variable

# Splitting into Training and Testing Sets (7:3 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0)

print("Training Set Shape:", X_train.shape, y_train.shape)
print("Testing Set Shape:", X_test.shape, y_test.shape)

# 4. Building a Logistic Regression Model
logistic_regressor = LogisticRegression()  # Creating the logistic
regression model
logistic_regressor.fit(X_train, y_train)  # Fitting the model

# Making predictions on the test set
y_pred = logistic_regressor.predict(X_test)

# 5. Evaluating the Model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

```python
print(f"Accuracy: {accuracy:.2f}")
print("Confusion Matrix:")
print(conf_matrix)

# Visualizing the Confusion Matrix
sn.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=['Not Purchased', 'Purchased'], yticklabels=['Not
Purchased', 'Purchased'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

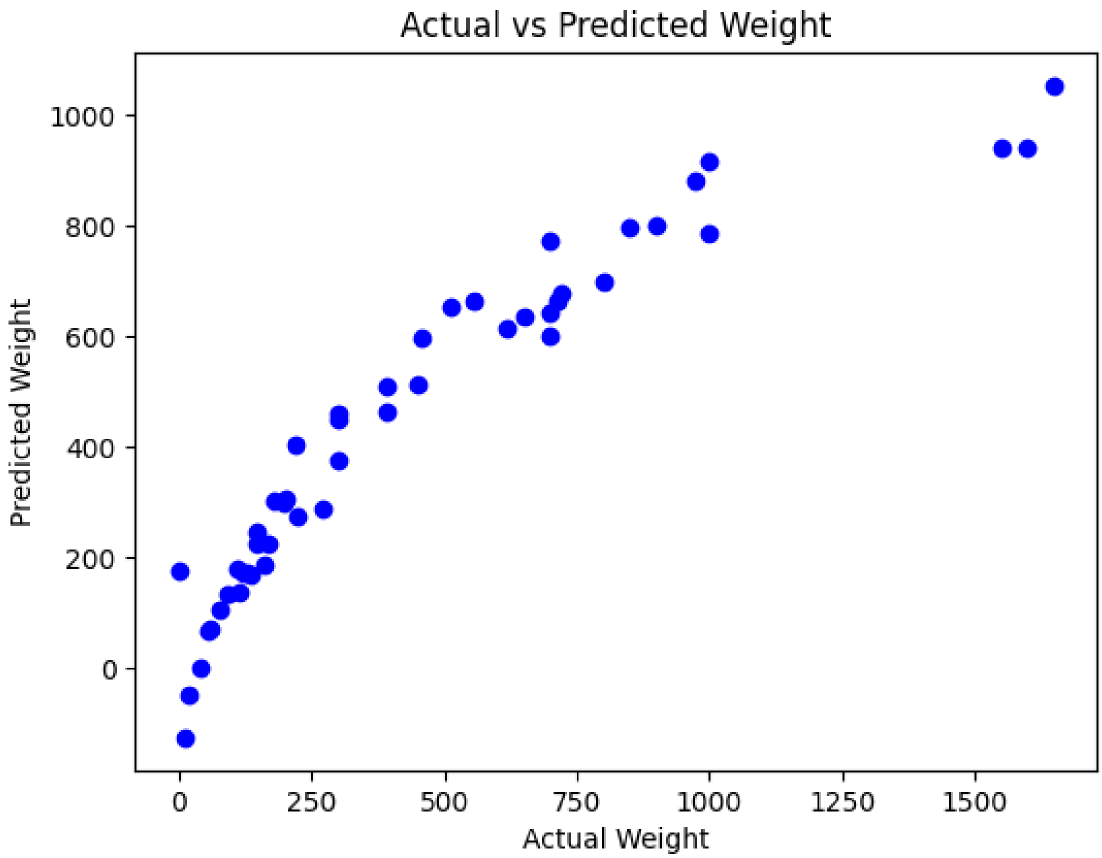
```
   User ID  Gender  Age  EstimatedSalary  Purchased
0        1    Male   36     93328.366843          0
1        2  Female   41    144401.286761          0
2        3  Female   19     34753.168976          1
3        4    Male   24     55389.816937          0
4        5  Female   48     30667.506897          0
Training Set Shape: (350, 3) (350,)
Testing Set Shape: (150, 3) (150,)
Accuracy: 0.71
Confusion Matrix:
[[106   0]
 [ 44   0]]
```




```
Question 4)

# 1. Import Libraries
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# 2. Load the Fish Dataset
# Assuming the dataset is downloaded and saved locally as 'Fish.csv'
data_path = 'C:\\Users\\ecs\\OneDrive\\Videos\\Documents\\Desktop\\
dataset\\Fish.csv'
data = pd.read_csv(data_path)
print(data.head())  # Previewing the first 5 rows

# 3. Splitting the Data into Independent and Target Variables
# Independent variables: Length1, Length2, Length3, Height, Width
# Target variable: Weight
X = data[['Length1', 'Length2', 'Length3', 'Height', 'Width']].values
y = data['Weight'].values
```

```python
# Splitting into Training and Testing Sets (7:3 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0)

print("Training Set Shape:", X_train.shape, y_train.shape)
print("Testing Set Shape:", X_test.shape, y_test.shape)

# 4. Building a Linear Regression Model
linear_regressor = LinearRegression()  # Creating the linear
regression model
linear_regressor.fit(X_train, y_train)  # Fitting the model

# Making predictions on the test set
y_pred = linear_regressor.predict(X_test)

# 5. Evaluating the Model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R2 Score: {r2:.2f}")

# Visualizing Actual vs Predicted
plt.scatter(y_test, y_pred, color="blue")
plt.xlabel('Actual Weight')
plt.ylabel('Predicted Weight')
plt.title('Actual vs Predicted Weight')
plt.show()
```

```
  Species  Weight  Length1  Length2  Length3   Height   Width
0   Bream   242.0     23.2     25.4     30.0  11.5200  4.0200
1   Bream   290.0     24.0     26.3     31.2  12.4800  4.3056
2   Bream   340.0     23.9     26.5     31.1  12.3778  4.6961
3   Bream   363.0     26.3     29.0     33.5  12.7300  4.4555
4   Bream   430.0     26.5     29.0     34.0  12.4440  5.1340
Training Set Shape: (111, 5) (111,)
Testing Set Shape: (48, 5) (48,)
Mean Squared Error: 32509.60
R2 Score: 0.81
```




```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("vipullrathod/fish-market")

print("Path to dataset files:", path)
```

```
Downloading from
https://www.kaggle.com/api/v1/datasets/download/vipullrathod/fish-
market?dataset_version_number=1...

100%|██████████| 2.38k/2.38k [00:00<00:00, 484kB/s]

Extracting files...
Path to dataset files: C:\Users\ecs\.cache\kagglehub\datasets\
vipullrathod\fish-market\versions\1
```

Question 5)

```
# 1. Import Libraries
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# 2. Load the Iris Dataset
from sklearn.datasets import load_iris
data = load_iris()
iris_df = pd.DataFrame(data=data.data, columns=data.feature_names)
iris_df['species'] = data.target

print(iris_df.head())  # Previewing the first 5 rows

# Viewing basic statistical details for each species
for species in range(3):
    print(f"Statistics for {data.target_names[species]}:")
    print(iris_df[iris_df['species'] == species].describe())
    print("\n")

# 3. Splitting the Data into Independent and Target Variables
X = iris_df.iloc[:, :-1].values  # Features: sepal and petal lengths
and widths
y = iris_df['species'].values   # Target: species

# Splitting into Training and Testing Sets (7:3 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0)

print("Training Set Shape:", X_train.shape, y_train.shape)
print("Testing Set Shape:", X_test.shape, y_test.shape)

# 4. Building a Logistic Regression Model
logistic_regressor = LogisticRegression(max_iter=200)  # Creating the
logistic regression model
logistic_regressor.fit(X_train, y_train)  # Fitting the model

# Making predictions on the test set
y_pred = logistic_regressor.predict(X_test)

# 5. Evaluating the Model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred,
target_names=data.target_names)

print(f"Accuracy: {accuracy:.2f}")
```

```python
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)

# Visualizing the Confusion Matrix
plt.figure(figsize=(8, 6))
sn.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=data.target_names, yticklabels=data.target_names)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2

   species
0        0
1        0
2        0
3        0
4        0
Statistics for setosa:
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count           50.00000         50.000000          50.000000
mean             5.00600          3.428000           1.462000
std              0.35249          0.379064           0.173664
min              4.30000          2.300000           1.000000
25%              4.80000          3.200000           1.400000
50%              5.00000          3.400000           1.500000
75%              5.20000          3.675000           1.575000
max              5.80000          4.400000           1.900000

       petal width (cm)  species
count         50.000000     50.0
mean           0.246000      0.0
std            0.105386      0.0
min            0.100000      0.0
25%            0.200000      0.0
```

```
50%             0.200000      0.0
75%             0.300000      0.0
max             0.600000      0.0


Statistics for versicolor:
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count          50.000000         50.000000          50.000000
mean            5.936000          2.770000           4.260000
std             0.516171          0.313798           0.469911
min             4.900000          2.000000           3.000000
25%             5.600000          2.525000           4.000000
50%             5.900000          2.800000           4.350000
75%             6.300000          3.000000           4.600000
max             7.000000          3.400000           5.100000

       petal width (cm)  species
count         50.000000     50.0
mean           1.326000      1.0
std            0.197753      0.0
min            1.000000      1.0
25%            1.200000      1.0
50%            1.300000      1.0
75%            1.500000      1.0
max            1.800000      1.0


Statistics for virginica:
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count           50.00000         50.000000          50.000000
mean             6.58800          2.974000           5.552000
std              0.63588          0.322497           0.551895
min              4.90000          2.200000           4.500000
25%              6.22500          2.800000           5.100000
50%              6.50000          3.000000           5.550000
75%              6.90000          3.175000           5.875000
max              7.90000          3.800000           6.900000

       petal width (cm)  species
count          50.00000     50.0
mean            2.02600      2.0
std             0.27465      0.0
min             1.40000      2.0
25%             1.80000      2.0
50%             2.00000      2.0
75%             2.30000      2.0
max             2.50000      2.0


Training Set Shape: (105, 4) (105,)
```

```
Testing Set Shape: (45, 4) (45,)
Accuracy: 0.98
Confusion Matrix:
[[16  0  0]
 [ 0 17  1]
 [ 0  0 11]]
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        16
  versicolor       1.00      0.94      0.97        18
   virginica       0.92      1.00      0.96        11

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```