EXPERIMENT 4::

# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# Load the California Housing dataset

from sklearn.datasets import fetch\_california\_housing

housing = fetch\_california\_housing()

# Create a DataFrame from the dataset

data = pd.DataFrame(housing.data, columns=housing.feature\_names)

data['MedHouseVal'] = housing.target

# Define the features (X) and the target (y)

X = data.drop('MedHouseVal', axis=1)

y = data['MedHouseVal']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a linear regression model

model = LinearRegression()

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

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

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

# Plotting the results

plt.scatter(y\_test, y\_pred)

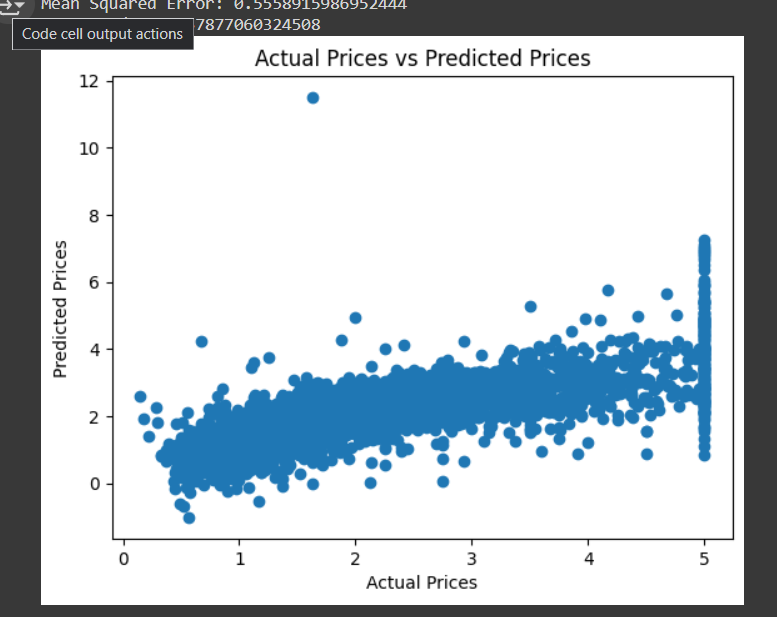
plt.xlabel("Actual Prices")

plt.ylabel("Predicted Prices")

plt.title("Actual Prices vs Predicted Prices")

plt.show()

OUTPUT:



EXPERIMENT 5::

# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, roc\_auc\_score

import matplotlib.pyplot as plt

# Load the Breast Cancer dataset

from sklearn.datasets import load\_breast\_cancer

data = load\_breast\_cancer()

# Create a DataFrame from the dataset

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

# Define the features (X) and the target (y)

X = df.drop('target', axis=1)

y = df['target']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a logistic regression model

model = LogisticRegression(max\_iter=10000)

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate 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)

roc\_auc = roc\_auc\_score(y\_test, model.predict\_proba(X\_test)[:, 1])

print(f"Accuracy: {accuracy}")

print(f"Confusion Matrix:\n{conf\_matrix}")

print(f"Classification Report:\n{class\_report}")

print(f"ROC AUC: {roc\_auc}")

# Plotting the ROC curve

fpr, tpr, \_ = roc\_curve(y\_test, model.predict\_proba(X\_test)[:, 1])

plt.plot(fpr, tpr, marker='.')

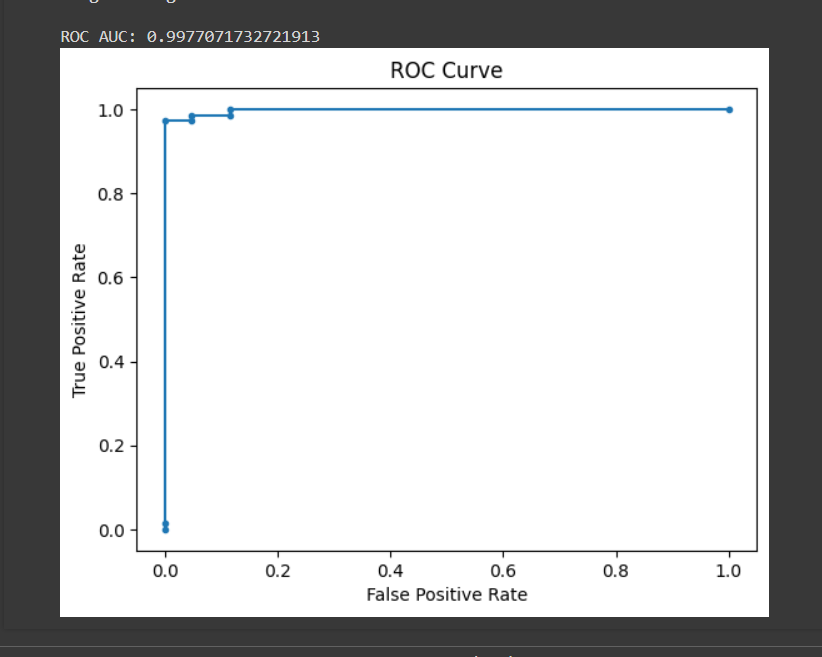
plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve")

plt.show()

OUTPUT:



1. EXPERIMENT 6::

# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

# Create a DataFrame from the dataset

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

data['target'] = iris.target

# Define the features (X) and the target (y)

X = data.drop('target', axis=1)

y = data['target']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a KNN model

knn = KNeighborsClassifier(n\_neighbors=3) # You can choose a different value for n\_neighbors

# Train the model

knn.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = knn.predict(X\_test)

# Evaluate the model's accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

B)# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

# Create a DataFrame from the dataset

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

data['target'] = iris.target

# Define the features (X) and the target (y)

X = data.drop('target', axis=1)

y = data['target']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Naive Bayes model

model = GaussianNB()

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

C)# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

# Create a DataFrame from the dataset

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

data['target'] = iris.target

# Define the features (X) and the target (y)

X = data.drop('target', axis=1)

y = data['target']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Logistic Regression model

model = LogisticRegression(max\_iter=200)

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

D)# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

# Create a DataFrame from the dataset

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

data['target'] = iris.target

# Define the features (X) and the target (y)

X = data.drop('target', axis=1)

y = data['target']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Decision Tree model

model = DecisionTreeClassifier(random\_state=42)

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

E)# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

# Create a DataFrame from the dataset

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

data['target'] = iris.target

# Define the features (X) and the target (y)

X = data.drop('target', axis=1)

y = data['target']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create an SVM model

model = SVC()

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

F)# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Load the Iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

# Create a DataFrame from the dataset

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

data['target'] = iris.target

# Define the features (X) and the target (y)

X = data.drop('target', axis=1)

y = data['target']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a Random Forest model

model = RandomForestClassifier(random\_state=42)

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")