Experiment No:1

AIM:

Create a numpy array and perform the following operations

Append values to the end of an array. Insert values into an array at a specified position. Delete elements from an array. Find unique elements in an array. Sort an array. Save an array to a text file. Load data from a text file into an array.

PROCEDURE

import numpy as np

my\_array = np.array([1, 2, 3, 4, 5])

my\_array = np.append(my\_array, [6, 7, 8])

my\_array = np.insert(my\_array, 2, 0)

my\_array = np.delete(my\_array, 3)

unique\_elements = np.unique(my\_array)

my\_array = np.sort(my\_array)

np.savetxt('my\_array.txt', my\_array)

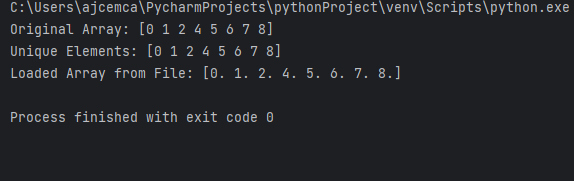
loaded\_array = np.loadtxt('my\_array.txt')

print("Original Array:", my\_array)

print("Unique Elements:", unique\_elements)

print("Loaded Array from File:", loaded\_array)

OUTPUT



Experiment No:2

AIM:

You have two NumPy arrays, arr1 and arr2, containing the following data:

arr1 = np.array([1, 2, 3, 4, 5])

arr2 = np.array([6, 7, 8, 9, 10])

Write NumPy code to perform the following operations: Add arr1 and arr2 to create a new array called result\_add. Multiply arr1 and arr2 to create a new array called result\_multiply. Calculate the mean of result\_add. Find the maximum value in result\_multiply.

PROCEDURE

import numpy as np

arr1 = np.array([1, 2, 3, 4, 5])

arr2 = np.array([6, 7, 8, 9, 10])

result\_add = arr1 + arr2

result\_multiply = arr1 \* arr2

mean\_result\_add = np.mean(result\_add)

max\_result\_multiply = np.max(result\_multiply)

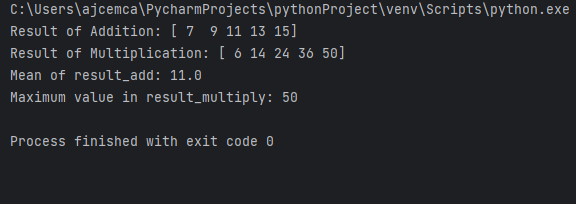
print("Result of Addition:", result\_add)

print("Result of Multiplication:", result\_multiply)

print("Mean of result\_add:", mean\_result\_add)

print("Maximum value in result\_multiply:", max\_result\_multiply)

OUTPUT



Experiment No:3

AIM: You have a NumPy array called grades that represents the scores of students in a class:

grades = np.array([85, 90, 78, 92, 88, 76, 95, 89, 84, 91])

Write NumPy code to answer the following questions: What is the average (mean) grade in the class? How many students scored above 90? Calculate the standard deviation of the grades.

PROCEDURE:

import numpy

grades = np.array([85, 90, 78, 92, 88, 76, 95, 89, 84, 91])

average\_grade = np.mean(grades)

above\_90\_count = np.sum(grades > 90)

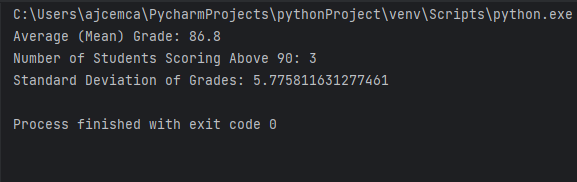
std\_deviation = np.std(grades)

print("Average (Mean) Grade:", average\_grade)

print("Number of Students Scoring Above 90:", above\_90\_count)

print("Standard Deviation of Grades:", std\_deviation)

OUTPUT



Experiment No:4

AIM:

Matrix Operations with NumPy

#Define the matrices

matrix\_A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

matrix\_B = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

Perform the following matrix operations: Add matrix\_A and matrix\_B element-wise to create a new matrix, matrix\_sum. Multiply matrix\_A and matrix\_B element-wise to create a new matrix, matrix\_product. Calculate the matrix product of matrix\_A and matrix\_B (dot product) and store it in matrix\_dot. Transpose matrix\_A and store it in matrix\_A\_transpose. Calculate the determinant of matrix\_B and store it in determinant\_B. Find the eigenvalues and eigenvectors of matrix\_A and store them in eigenvalues\_A and eigenvectors\_A. Find SVD of a matrix

PROCEDURE:

import numpy as np

matrix\_A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

matrix\_B = np.array([[9, 8, 7],[6, 5, 4], [3, 2, 1]])

matrix\_sum = matrix\_A + matrix\_B

matrix\_product = matrix\_A \* matrix\_B

matrix\_dot = np.dot(matrix\_A, matrix\_B)

matrix\_A\_transpose = matrix\_A.T

determinant\_B = np.linalg.det(matrix\_B)

eigenvalues\_A, eigenvectors\_A = np.linalg.eig(matrix\_A)

U, S, Vt = np.linalg.svd(matrix\_A)

print("Matrix Sum:")

print(matrix\_sum)

print("\nMatrix Element-wise Product:")

print(matrix\_product)

print("\nMatrix Dot Product:")

print(matrix\_dot)

print("\nMatrix A Transpose:")

print(matrix\_A\_transpose)

print("\nDeterminant of Matrix B:", determinant\_B)

print("\nEigenvalues of Matrix A:")

print(eigenvalues\_A)

print("\nEigenvectors of Matrix A:")

print(eigenvectors\_A)

print("\nSingular Value Decomposition (SVD) of Matrix A:")

print("U:")

print(U)

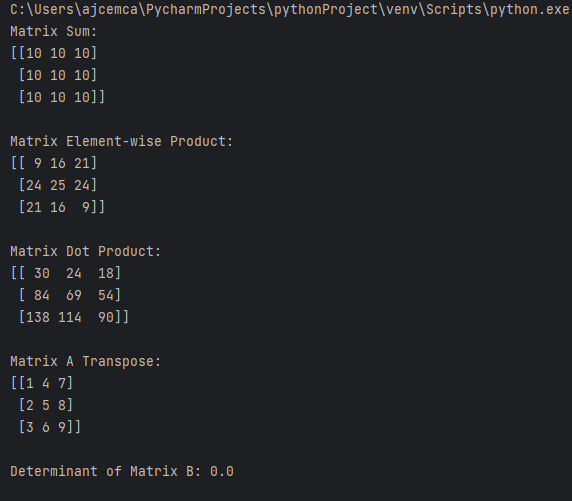
print("S:")

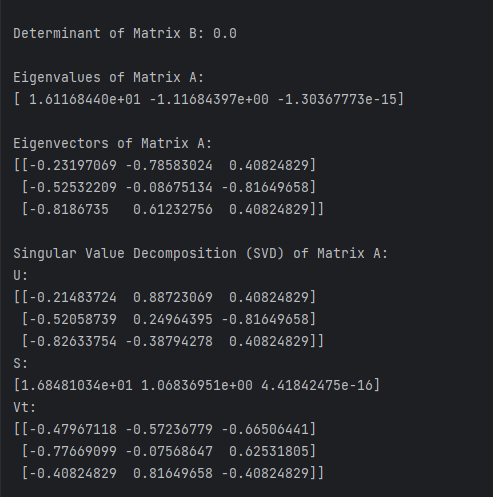
print(S)

print("Vt:")

print(Vt)

OUTPUT





Experiment No:5

AIM:You have a CSV file named "sales\_data.csv" containing sales data with columns for "Date," "Product," "Quantity," and "Revenue." Load this data using Pandas and answer the following questions: How many rows and columns are there in the dataset? What is the total revenue for all the sales?

PROCEDURE:

import pandas as pd

df = pd.read\_csv("sales\_data.csv")

num\_rows, num\_columns = df.shape

total\_revenue = df["Revenue"].sum()

print("Number of Rows:", num\_rows)

print("Number of Columns:", num\_columns)

print("Total Revenue for All Sales:", total\_revenue)

OUTPUT

Experiment No:6

AIM:

You have a DataFrame called "student\_data" with columns "Student\_ID," "Name," "Age," and "GPA." Perform the following operations using Pandas: Filter and display the rows of students who are 20 years old or older

. Calculate the average GPA of the students in the DataFrame.

Sort the DataFrame in descending order of GPA and display the top 5 students with the highest GPAs. Group the students by their ages and calculate the average GPA for each age group.

Reference: HISTOGRAM

from matplotlib

import pyplot as plt

import numpy as np

# Creating dataset a = np.array([22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27])

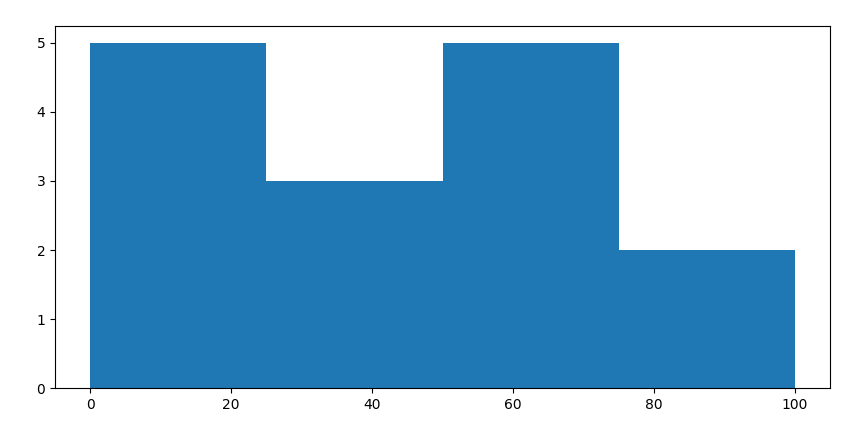
# Creating histogram fig,

ax = plt.subplots(figsize =(10, 7))

ax.hist(a, bins = [0, 25, 50, 75, 100])

# Show plot plt.show()

OUTPUT



Experiment No:7

AIM:Histogram and Quartile Plot: Objective: Visualize the distribution of a univariate dataset(consists of observations or measurements on a single variable or characteristic) and analyze its quartiles. Question: Use a dataset of your choice (e.g., exam scores of students, employee salaries, or any other numerical data). Create a histogram to visualize the data's distribution. Afterward, plot quartiles (e.g., Q1, Q2, Q3) on the same graph. Answer the following questions: What does the histogram reveal about the data's distribution? How do the quartiles relate to the histogram? Are there any outliers in the data, and if so, how do they affect the quartiles? Output: Provide the histogram and quartile plot along with a written analysis.

PROCEDURE

import matplotlib.pyplot as plt

import numpy as np

exam\_scores = [60, 72, 78, 85, 90, 92, 94, 95, 98, 100, 102, 105, 110]

plt.hist(exam\_scores, bins=10, edgecolor='k', alpha=0.7)

plt.xlabel('Exam Scores')

plt.ylabel('Frequency')

plt.title('Histogram of Exam Scores')

plt.grid(True)

q1 = np.percentile(exam\_scores, 25)

q2 = np.percentile(exam\_scores, 50)

q3 = np.percentile(exam\_scores, 75)

plt.axvline(q1, color='r', linestyle='--', label='Q1')

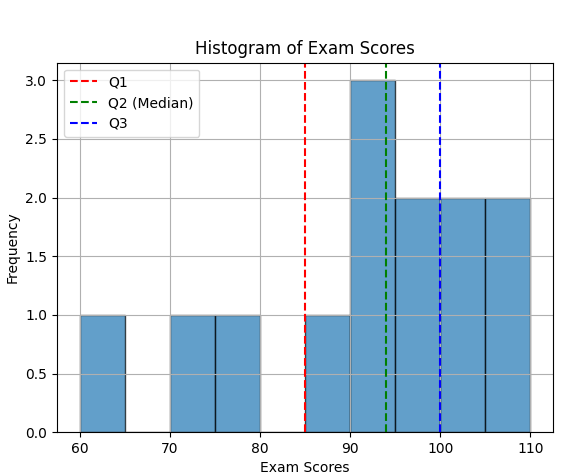
plt.axvline(q2, color='g', linestyle='--', label='Q2 (Median)')

plt.axvline(q3, color='b', linestyle='--', label='Q3')

plt.legend()

plt.show()

OUTPUT



Experiment No:8

AIM:Distribution Chart and Scatter Plot: Objective

: Explore the relationship between two variables and visualize their distributions.

Question: Choose a dataset that contains two numerical variables (e.g., income vs. education level, temperature vs. ice cream sales). Create a distribution chart for each variable and a scatter plot to visualize their relationship. Answer the following questions: What do the distribution charts reveal about each variable? Is there a correlation between the two variables based on the scatter plot? Can you identify any patterns or trends in the data? Output: Present the distribution charts, scatter plot, and your observations in a report.

PROCEDURE

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

data = {

'Income': [30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000],

'Education\_Level': [10, 12, 8, 14, 16, 12, 18, 20, 14, 22]

}

df = pd.DataFrame(data)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

sns.histplot(df['Income'], kde=True)

plt.xlabel('Income')

plt.ylabel('Frequency')

plt.title('Distribution of Income')

plt.subplot(1, 2, 2)

sns.histplot(df['Education\_Level'], kde=True)

plt.xlabel('Education Level')

plt.ylabel('Frequency')

plt.title('Distribution of Education Level')

plt.tight\_layout()

# Scatter plot to visualize the relationship

plt.figure(figsize=(6, 6))

sns.scatterplot(x='Income', y='Education\_Level', data=df)

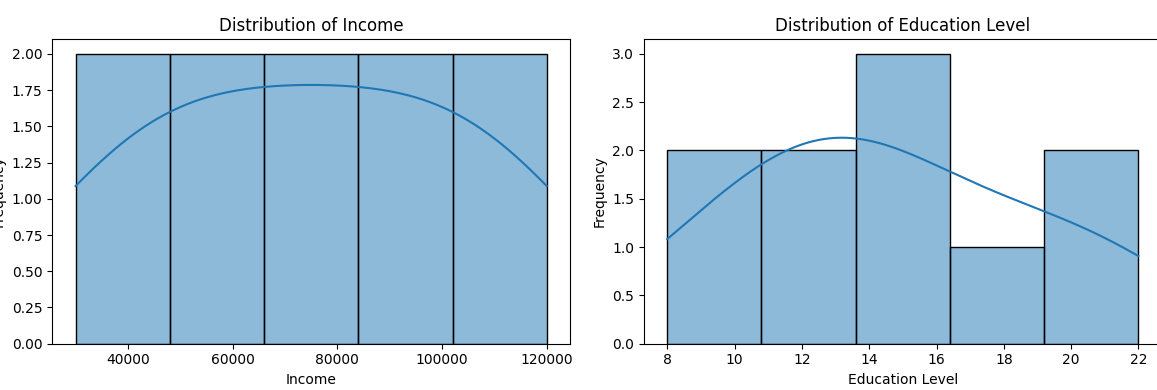
plt.xlabel('Income')

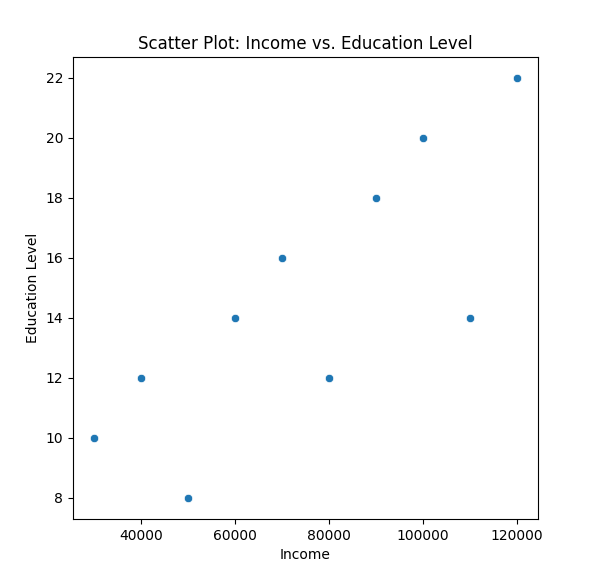
plt.ylabel('Education Level')

plt.title('Scatter Plot: Income vs. Education Level')

plt.show()

OUTPUT





Experiment No:9

AIM:Bubble Chart and Density Chart:

Objective: Visualize multivariate data using bubble and density charts.

Question: Select a dataset with at least three numerical variables (e.g., population, income, and education level by city). Create a bubble chart that represents the data by using bubble sizes and colors to encode information. Additionally, create a density chart (e.g., a 2D density plot) to show the concentration of data points. Answer the following questions: How does the bubble chart help in visualizing multivariate data? What insights can you gain from the density chart in terms of data concentration? Are there any interesting patterns or outliers in the data? Output: Share the bubble chart, density chart, and an interpretation of the visualizations.

PROCEDURE

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

np.random.seed(42)

n\_samples = 100

data = {

'City': [f'City {i}' for i in range(n\_samples)],

'Population': np.random.randint(10000, 200000, n\_samples),

'Income': np.random.randint(30000, 80000, n\_samples),

'Education\_Level': np.random.randint(8, 20, n\_samples),

}

df = pd.DataFrame(data)

plt.figure(figsize=(10, 6))

sns.scatterplot(data=df, x='Income', y='Education\_Level', size='Population', sizes=(10, 300),

hue='Population', palette='coolwarm', alpha=0.7)

plt.xlabel('Income')

plt.ylabel('Education Level')

plt.title('Bubble Chart: Income vs. Education Level')

plt.legend(title='Population')

plt.grid(True)

plt.figure(figsize=(10, 6))

sns.kdeplot(data=df, x='Income', y='Education\_Level', fill=True, cmap='viridis', levels=10)

plt.xlabel('Income')

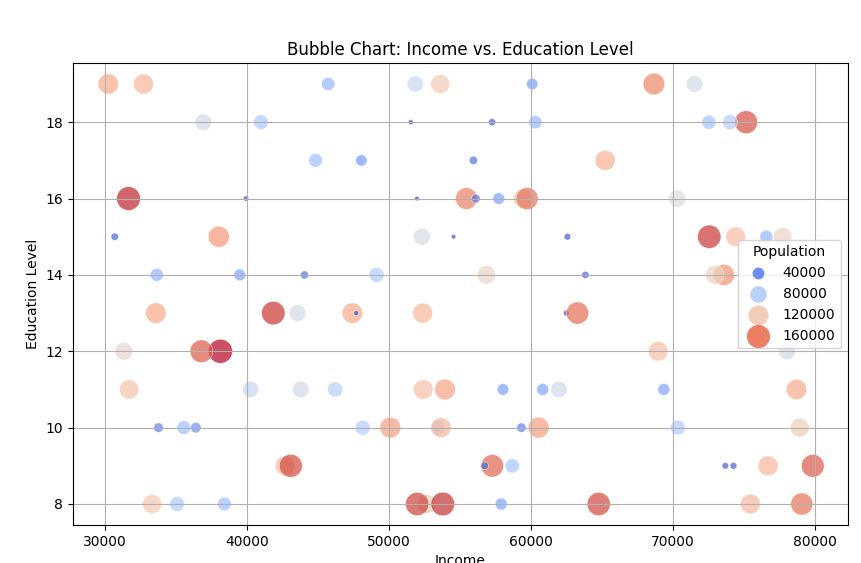
plt.ylabel('Education Level')

plt.title('2D Density Plot: Income vs. Education Level')

plt.grid(True)

plt.show()

OUTPUT



EXPERIMENT 10

AIM : **K-NN Classificationimport numpy as np**

CO1 :

PROGRAM :

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

from sklearn.neighbors import KNeighborsClassifier

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

X = np.random.rand(100, 2) # Features

y = np.random.randint(0, 2, 100) # Target labels (binary classification)

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

knn = KNeighborsClassifier(n\_neighbors=3)

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

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

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

print(conf\_matrix)

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

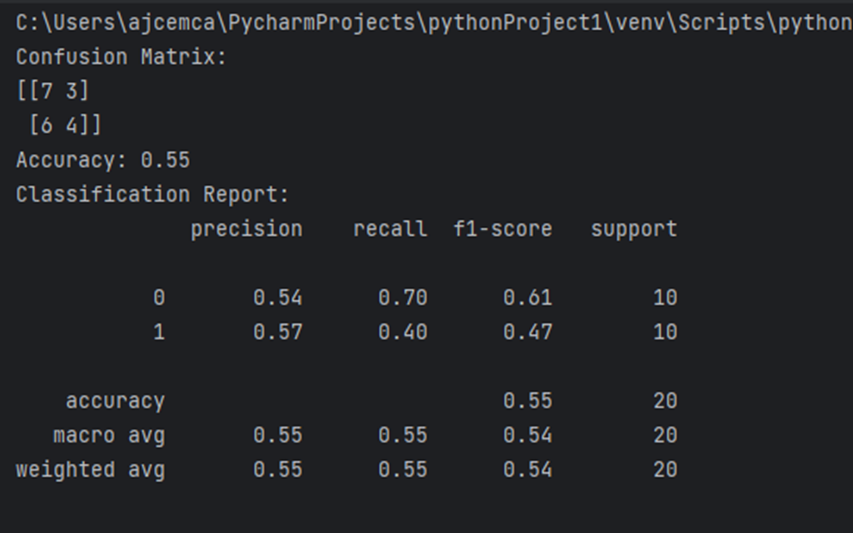
print("Accuracy:", accuracy)

class\_report = classification\_report(y\_test, y\_pred)

print("Classification Report:")

print(class\_report)

OUTPUT



EXPERIMENT 11

AIM : Imagine you want to predict a student's final exam score based on the number of hours they study. You have collected data for several students and their exam scores and study hours. Now, you want to build a simple linear regression model to predict a study

import matplotlib.pyplot as plt

import numpy as np

x=np.array([2,3,4,5,6])

y=np.array([60,75,80,90,95])

xi=np.mean(x)

print("Mean of study hours",xi)

yi=np.mean(y)

print("Mean of exam scores",yi)

m=np.sum((x-xi)\*(y-yi))/np.sum((x-xi)\*\*2)

b=yi-m\*xi

def predict\_exam\_score(hours):

return m \* hours + b

predicted\_score = predict\_exam\_score(4)

print("predicted hours of 4 :{predicted\_score} = ",predicted\_score)

regression\_line=m\*x+b

plt.scatter(x,y, color='blue',label='data')

plt.plot(x,regression\_line,color='red',label='regression line')

plt.scatter(4,predicted\_score, color='yellow',label='predicted point')

plt.xlabel(x)

plt.ylabel(y)

plt.legend()

plt.show()

● By using csv

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

#using array

#x=np.array([2,3,4,5,6])

#y=np.array([60,75,80,90,95])

#using csv file

df=pd.read\_csv("studyexam.csv")

x=df['study hours']

y=df['exam score']

xi=np.mean(x)

print("Mean of study hours",xi)

yi=np.mean(y)

print("Mean of exam scores",yi)

m=np.sum((x-xi)\*(y-yi))/np.sum((x-xi)\*\*2)

b=yi-m\*xi

def predict\_exam\_score(hours):

return m \* hours + b

predicted\_score = predict\_exam\_score(4)

print("predicted hours of 4 :{predicted\_score} = ",predicted\_score)

regression\_line=m\*x+b

plt.scatter(x,y, color='blue',label='data')

plt.plot(x,regression\_line,color='red',label='regression line')

plt.scatter(4,predicted\_score, color='yellow',label='predicted point')

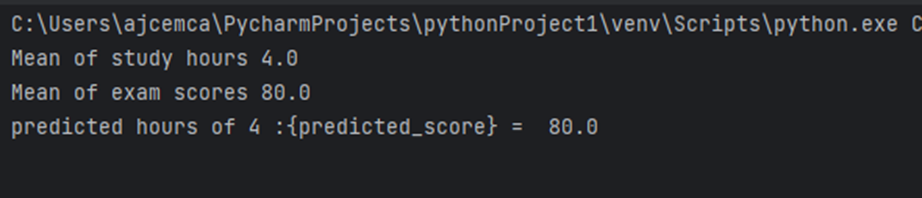
plt.xlabel(x)

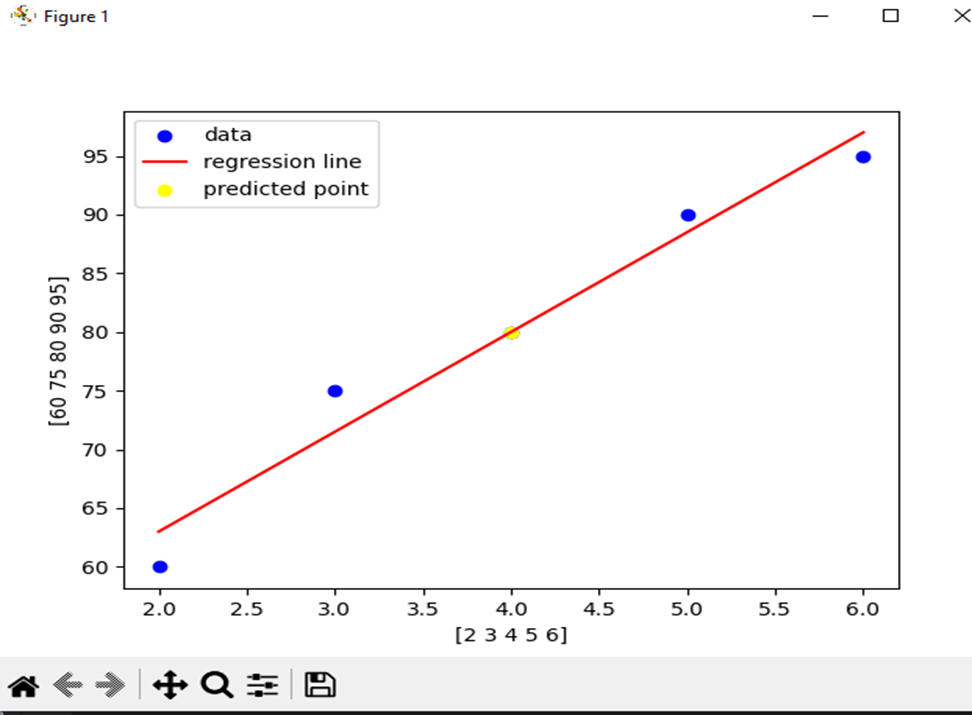
plt.ylabel(y)

plt.legend()

plt.show()

OUTPUT





EXPERIMENT 12

AIM : Multiple linear regression

import pandas as pd

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

data = pd.read\_csv("details.csv")

x = data[["study\_hours", "previous\_testscore", "extracaricular\_activities"]]

y = data["exam\_score"]

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

model = LinearRegression()

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

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

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

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

print("Mean Squared Error:", mse)

print("R-squared (R^2):", r2)

print("Coefficients (Slopes):", model.coef\_)

print("Intercept:", model.intercept\_)

#plt.scatter(x,y, color='blue',label='data')

#plt.scatter(y\_pred, color='yellow',label='predicted point')

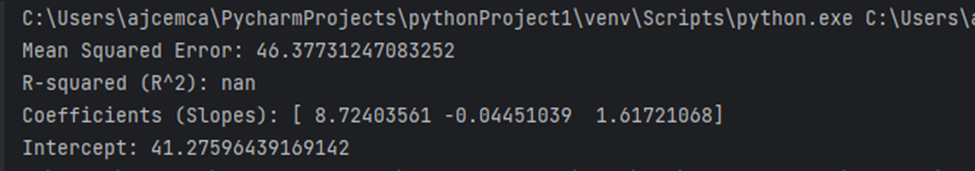
#plt.xlabel(x)

#plt.ylabel(y)

#plt.legend()

#plt.show()

OUTPUT



EXPERIMENT 13

AIM : DECISION TREE

from sklearn.datasets import load\_iris

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

from sklearn.tree import DecisionTreeClassifier

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

import matplotlib.pyplot as plt

from sklearn import tree

iris = load\_iris()

X = iris.data

y = iris.target

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

predi = DecisionTreeClassifier()

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

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

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

print(f'Accuracy: {accuracy:.2f}')

print(classification\_report(y\_test, y\_pred))

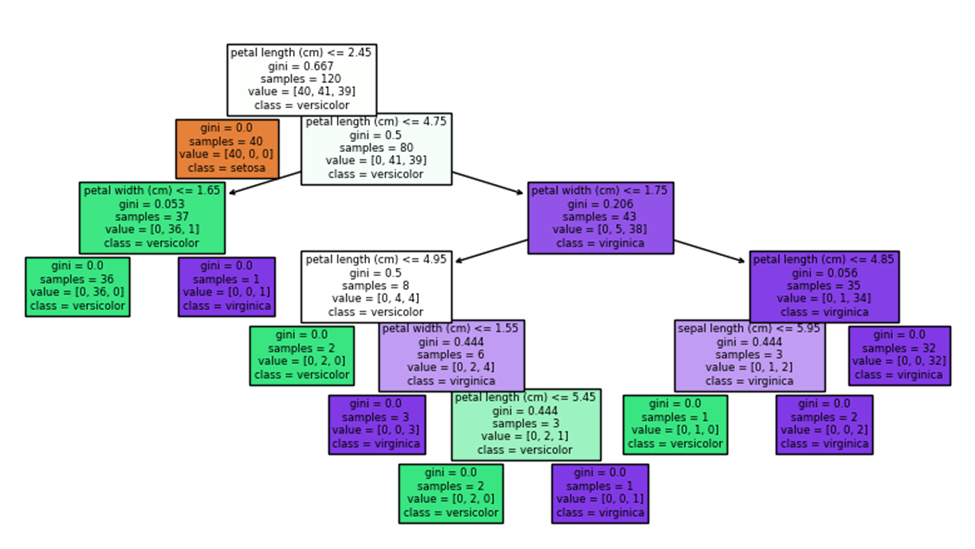
print(f'Confusion Matrix:\n{confusion\_matrix(y\_test, y\_pred)}')

plt.figure(figsize=(10, 5))

tree.plot\_tree(predi, feature\_names=iris.feature\_names, class\_names=iris.target\_names, filled=True)

plt.show()

OUTPUT



Decesiontree method2

import pandas as pd

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

from sklearn.tree import DecisionTreeClassifier

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

import matplotlib.pyplot as plt

from sklearn import tree

iris=pd.read\_csv("irisdata.csv")

X=iris[['no\_of\_celebreties','budget']]

y=iris['Estimated\_success']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=2)

predi = DecisionTreeClassifier()

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

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

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

print(f'Accuracy: {accuracy:.2f}')

print(classification\_report(y\_test, y\_pred))

print(f'Confusion Matrix:\n{confusion\_matrix(y\_test, y\_pred)}')

plt.figure(figsize=(10, 5))

tree.plot\_tree(predi, feature\_names=['no\_of\_celebreties','budget'], class\_names=iris['Estimated\_success'].unique(), filled=True)

plt.show()

OUTPUT

