20MCA241 - DATA SCIENCE

LABORATORY RECORD

Submitted in partial fulfilment of the requirements for the award of

Masters of Computer Applications

At

COLLEGE OF ENGINEERING POONJAR

Managed by I.H.R.D., A Govt. of Kerala undertaking

(Affiliated to APJ Abdul Kalam Technological University)

SUBMITTED BY

JEETHU DANIEL(PJR24MCA-2009)

Department of Computer Science

COLLEGE OF ENGINEERING POONJAR

COLLEGE OF ENGINEERING POONJAR

Managed by I.H.R.D., A Govt. of Kerala undertaking

(Affiliated to APJ Abdul Kalam Technological University)

CERTIFICATE

Certified that this is a Bonafide record of practical work done in Data Science Lab (20MCA241) Laboratory by **JEETHU DANIEL** Reg No. **PJR24MCA-2009** of College of Engineering, Poonjar during the academic year 2024- 2026.

Dr. Annie Julie Joseph   **Mary Treesa Thomas**

Head of the Department   **Asst.Professor ,CSE**

Submitted to the University Examination held on:

INTERNAL EXAMINER   **EXTERNAL EXAMINER**

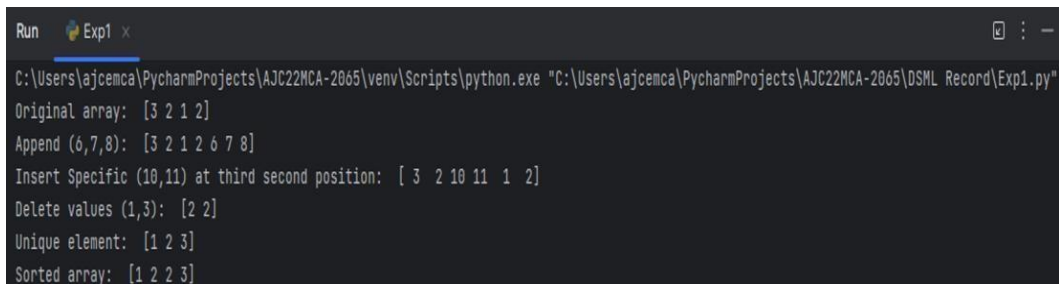# INDEX

PROGRAM - 1

AIM

Create a numpy array and perform the following operations

1. Append values to the end of an array.

2. 2.Insert values into an array at a specified position.

3. Delete elements from an array.

4. Find unique elements in an array.

5. Sort an array.

SOURCE CODE

```python
import numpy as np
array = np.array([3, 2, 1, 2])
print("Original array: ", array)
print("Append (6,7,8): ",np.append(array, [6, 7, 8]))
print("Insert Specific (10,11) at third second position: ", np.insert(array, 2, [10,11]))
print("Delete values (1,3): ", np.delete(array, [0, 2]))
print("Unique element: ", np.unique(array))
print("Sorted array: ", np.sort(array))
```

OUTPUT

```
Run    Exp1 x
C:\Users\ajcemca\PycharmProjects\AJC22MCA-2065\venv\Scripts\python.exe "C:\Users\ajcemca\PycharmProjects\AJC22MCA-2065\DSML Record\Exp1.py"
Original array:  [3 2 1 2]
Append (6,7,8):  [3 2 1 2 6 7 8]
Insert Specific (10,11) at third second position:  [ 3  2 10 11  1  2]
Delete values (1,3):  [2 2]
Unique element:  [1 2 3]
Sorted array:  [1 2 2 3]
```

RESULT

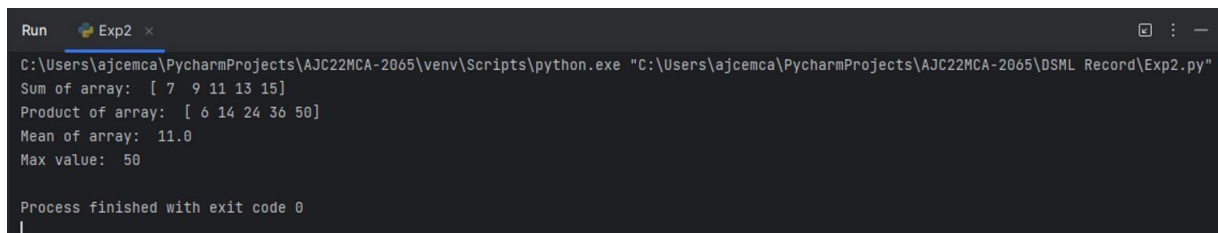The program has been executed successfully and output is obtained.

PROGRAM - 2

AIM

You have two NumPy arrays, arr1 and arr2.Write NumPy code to perform the following operations:

1. Add arr1 and arr2 to create a new array called result_add.

2. Multiply arr1 and arr2 to create a new array called result_multiply.

3. Calculate the mean of result_add.

4. Find the maximum value in result_multiply.

SOURCE CODE

```python
import numpy as np
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([6, 7, 8, 9, 10])
result_add = arr1 + arr2
print("Sum of array: ",result_add)
result_multiply = arr1 * arr2
print("Product of array: ", result_multiply)
print("Mean of array: ", np.mean(result_add))
print("Max value: ", np.max(result_multiply))
```

OUTPUT

```
Run    Exp2  ×
C:\Users\ajcemca\PycharmProjects\AJC22MCA-2065\venv\Scripts\python.exe "C:\Users\ajcemca\PycharmProjects\AJC22MCA-2065\DSML Record\Exp2.py"
Sum of array:  [ 7  9 11 13 15]
Product of array:  [ 6 14 24 36 50]
Mean of array:  11.0
Max value:  50

Process finished with exit code 0
```

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 3

AIM

You have a NumPy array called grades that represents the scores of students in a class:
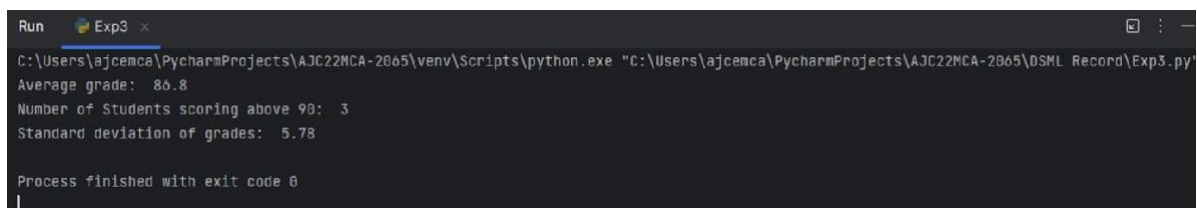
Write NumPy code to answer the following questions:

1. What is the average (mean) grade in the class?
2. How many students scored above 90?
3. Calculate the standard deviation of the grades.

SOURCE CODE

```
import numpy as np
grades = np.array([85, 90, 78, 92, 88, 76, 95, 89, 84, 91])
print("Average grade: ", np.mean(grades))
filter_grade = grades[grades > 90]
print("Number of Students scoring above 90: ", len(filter_grade))
std_grade = np.std(grades)
print("Standard deviation of grades: ",np.round(std_grade,decimals=2))
```

OUTPUT

```
Run    Exp3 ×
C:\Users\ajcemca\PycharmProjects\AJC22MCA-2865\venv\Scripts\python.exe "C:\Users\ajcemca\PycharmProjects\AJC22MCA-2865\DSML Record\Exp3.py"
Average grade:  86.8
Number of Students scoring above 90:  3
Standard deviation of grades:  5.78

Process finished with exit code 0
```

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 4

AIM

Perform the following matrix operations:

1. Add matrix_A and matrix_B element-wise to create a new matrix, matrix_sum.
2. Multiply matrix_A and matrix_B element-wise to create a new matrix, matrix_product.
3. Calculate the matrix product of matrix_A and matrix_B (dot product) and store it in matrix_dot.
4. Transpose matrix_A and store it in matrix_A_transpose.

Print the results of each operation.

SOURCE CODE
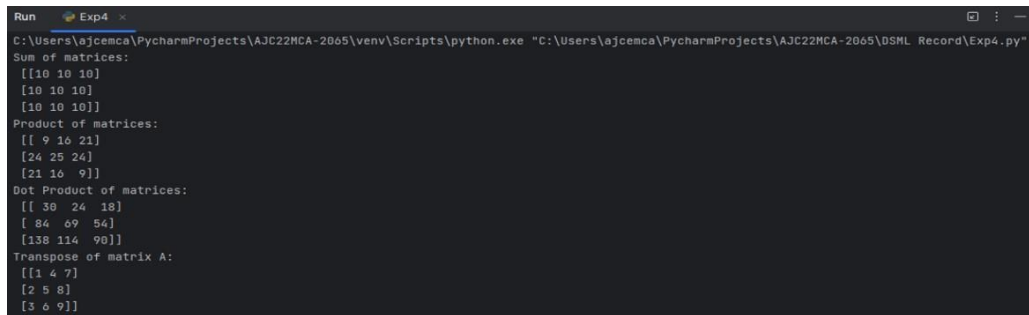
```python
import numpy as np
matrix_A = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
matrix_B = np.array([[9, 8, 7],[6, 5, 4],[3, 2, 1]])
matrix_sum = matrix_A + matrix_B
print("Sum of matrices: \n", matrix_sum)
matrix_product = matrix_A * matrix_B
print("Product of matrices: \n", matrix_product)
matrix_dot = np.dot(matrix_A, matrix_B)
print("Dot Product of matrices: \n",matrix_dot)
matrix_A_transpose = np.transpose(matrix_A)
print("Transpose of matrix A: \n", matrix_A_transpose)
```

OUTPUT

```
Run    Exp4  ×                                                                      ⊡  :  —
C:\Users\ajcemca\PycharmProjects\AJC22MCA-2065\venv\Scripts\python.exe "C:\Users\ajcemca\PycharmProjects\AJC22MCA-2065\DSML Record\Exp4.py"
Sum of matrices:
[[10 10 10]
 [10 10 10]
 [10 10 10]]
Product of matrices:
[[ 9 16 21]
 [24 25 24]
 [21 16  9]]
Dot Product of matrices:
[[ 30  24  18]
 [ 84  69  54]
 [138 114  90]]
Transpose of matrix A:
[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 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:

1. How many rows and columns are there in the dataset?
2. What is the total revenue for all the sales?

SOURCE CODE

```
import pandas as pd
 df = pd.read_csv("sales_data.csv")
print(df.head())
print("Number of Rows:",len(df))
print("Number ofcolumns:",len(df.columns))
print("Total revenue:",sum(df['revenue']))
//sales_data.csv date,product,quantity,revenue
2024-05-09,book,5,500
2024-07-12,pencil,5,10
2024-03-11,eraser,12,50
2024-08-23,scale,6,30
2024-11-23,pen,20,200
```

OUTPUT

```
/home/cep/PycharmProjects/pythonProject/.venv/bin/python /home/cep/Desktop/pythonProject/files.py
        date  product   quantity   revenue
0   2024-05-09     book          5       500
1   2024-07-12   pencil          5        10
2   2024-03-11   eraser         12        50
3   2024-08-23    scale          6        30
4   2024-11-23      pen         20       200
Number of Rows: 5
Number of columns: 4
Total revenue: 790

Process finished with exit code 0
```

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 6

AIM

You have a DataFrame called "student_data" with columns "Student_ID," "Name," "Age," and "GPA." Perform the following operations using Pandas:

1. Filter and display the rows of students who are 20 years old or older.
2. Calculate the average GPA of the students in the DataFrame.
3. Sort the DataFrame in descending order of GPA and display the top 5 students with the highest GPAs.
4. Group the students by their ages and calculate the average GPA for each age group.

SOURCE CODE

```
import pandas as pd
df = pd.read_csv("students_data.csv")
print(df)
print("no of students who are 20 years old or older")
age_df=df[df['age'] >=20]
print(age_df)
avggpa=df['GPA'].mean()
print("Average GPA: ")
print(avggpa)
sorted_df=df.sort_values(by='GPA',ascending=False)
print("descending order of GPA: ")
print(sorted_df)
top_five=sorted_df.head(5)
print("Top 5 students with highest GPA: ")
print(top_five)
avg_grp=df.groupby('age')['GPA'].mean().reset_index()
print("Average GPA of Each age group : ")
print(avg_grp)
//student_data.csv student_id,name,age,GPA
1001,Amal,23,9.8
```

1002,athul,18,8.8

1003,mathew,20,7.9

1004,thoma,21,8.9

1005,aby,19,6.8

1006,john,20,6.9

1007,robin,22,9.5

OUTPUT

```
/home/cep/PycharmProjects/pythonProject/.venv/bin/python /home/cep/Desktop/pythonProject/p6_student.py
   student_id    name  age  GPA
0        1001    Amal   23  9.8
1        1002   athul   18  8.8
2        1003  mathew   20  7.9
3        1004   thoma   21  8.9
4        1005     aby   19  6.8
5        1006    john   20  6.9
6        1007   robin   22  9.5
no of students who are 20 years old or older:
   student_id    name  age  GPA
0        1001    Amal   23  9.8
2        1003  mathew   20  7.9
3        1004   thoma   21  8.9
5        1006    john   20  6.9
6        1007   robin   22  9.5
Average GPA:
8.37142857142857
descending order of GPA:
   student_id    name  age  GPA
0        1001    Amal   23  9.8
6        1007   robin   22  9.5
3        1004   thoma   21  8.9
1        1002   athul   18  8.8
2        1003  mathew   20  7.9
5        1006    john   20  6.9
4        1005     aby   19  6.8
```

```
Top 5 students with highest GPA:
    student_id       name  age   GPA
0         1001       Amal   23   9.8
6         1007      robin   22   9.5
3         1004      thoma   21   8.9
1         1002      athul   18   8.8
2         1003     mathew   20   7.9
Average GPA of Each age group :
    age   GPA
0    18   8.8
1    19   6.8
2    20   7.4
3    21   8.9
4    22   9.5
5    23   9.8

Process finished with exit code 0
```

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 7

AIM

Create scatter plot for the below data:(use Scatter function)

Product	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Affordable Segment	173	153	195	147	120	144	148	109	174	130	172	131
Luxuary Segment	189	189	105	112	173	109	151	197	174	145	177	161
Super Luxury Segment	185	185	126	134	196	153	112	138	200	145	167	110

Create scatter plot for each Segment with following properties within one graph

- X Label -Month of Year with font size 18
- Y Label-Sales of Segments
- Title-Sales Data
- Color for Affordable segment-pink
- Color For Luxury segment

SOURCE CODE

```
import matplotlib.pyplot as plt
import numpy as np
month=np.array(['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'])
AS=np.array([173,153,195,147,120,144,148,109,174,130,172,131])
LS=np.array([189,189,105,112,173,109,151,197,174,145,177,161])
SLS=np.array([185,185,126,134,196,153,112,133,200,145,167,110])
plt.xlabel("Months of Year",fontsize=18)
plt.ylabel('Sales Of Segments')
plt.title('Sales Data')
plt.title('ANJIMA C P\n MCA 2024-2026',loc='right')
plt.scatter(month,AS,label='Affordable Segment',color='pink')
plt.scatter(month,LS,label=' Luxury Segment',color='blue')
plt.scatter(month,SLS,label='Super Luxury Segment',color='blue')
```

plt.savefig("months.png")

plt.show()

OUTPUT
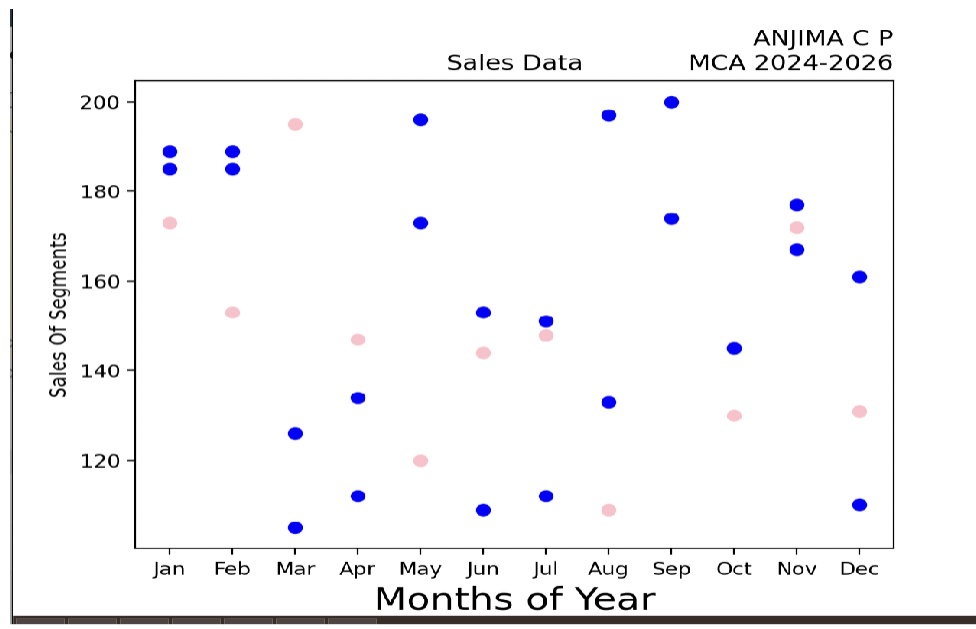
RESULT

The program has been executed successfully and output is obtained.

PROGRAM – 8

AIM

100 students were asked what their primary mode of transport for getting to school was.The results of this survey are recorded in the table below. Construct a bar graph representing this information.

| Mode of Transport	Frequency
Walking	29
Cycling	15
Car	35
Bus	18
Train	3

Create a bar graph

- X axis-mode of Transport and Y axis -frequency
- Provide appropriate labels and title
- Width.1,color blue

SOURCE CODE

```
import matplotlib.pyplot as plt
import numpy as np
mode_transport=np.array(["Walking",'Cycling','Car','Bus','Train'])
feq=np.array([29,15,35,18,3])
plt.xlabel('Mode of Transport')
plt.ylabel('Frequency')
plt.title('MCA 2024-2026',loc='right')
plt.bar(mode_transport,feq,width=0.1,color='blue')
plt.savefig("Transport.png")
plt.show()
```

OUTPUT
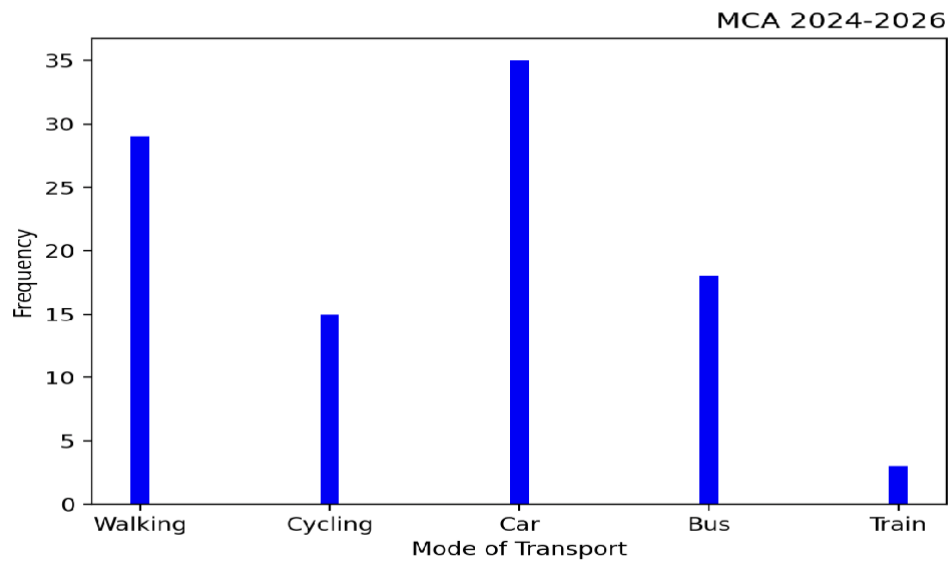
RESULT

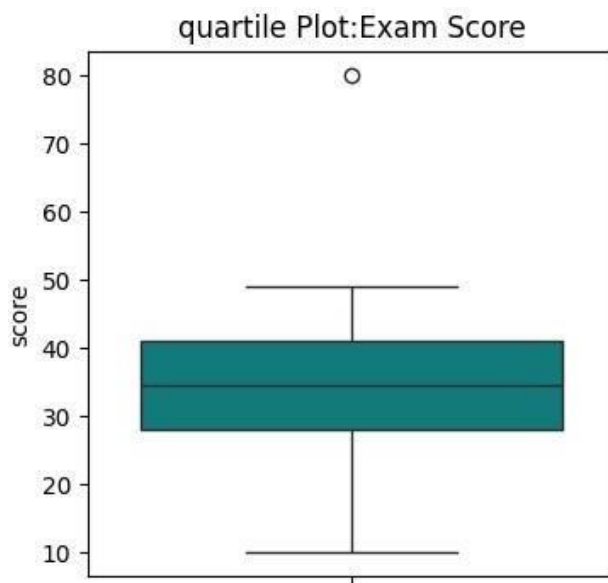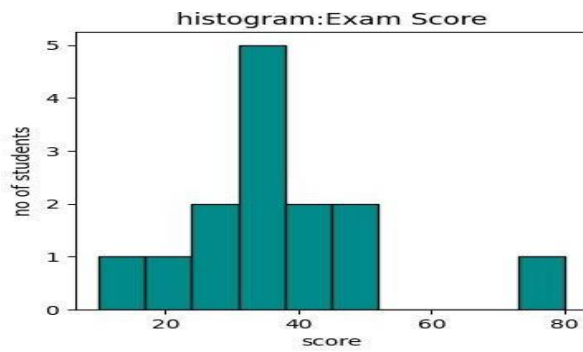The program has been executed successfully and output is obtained.

PROGRAM - 9

AIM

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.

SOURCE CODE

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
marks=np.array([10,18,34,37,33,38,24,80,45,49,27,31,35,42])
fig, ax=plt.subplots(figsize=(4,4))
ax.hist(marks, color="darkcyan",ec="black",lw=1)
plt.title('histogram:Exam Score')
plt.ylabel('no of students')
plt.xlabel('score')
plt.savefig('histogram.png')
plt.figure(figsize=(4,4))
sns.boxplot(y=marks,color="darkcyan")
plt.title('quartile Plot:Exam Score')
plt.ylabel('score')
plt.savefig('quartile_plot.png')
```

OUTPUT

histogram:Exam Score

quartile Plot:Exam Score

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 10

AIM

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.

SOURCE CODE

```
import matplotlib
matplotlib.use('Agg')  # Use non interactive backend
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Define the data
x_values = [1, 2, 3, 4, 5]
y_values = [10, 15, 13, 18, 20]
data_values = [10, 15, 13, 18, 20]

# Create DataFrame
 df = pd.DataFrame({'x': x_values, 'y': y_values, 'value':data_values})

# Create a pivot table for the heatmap
heatmap_data = df.pivot_table(index='x', columns='y', values='value')

# Plot the heatmap
fig, ax = plt.subplots(figsize=(10, 6))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu',cbar=True)
plt.title('Heatmap Example')
plt.savefig('heatmap_example.png')  # Save to file
plt.close()


# Scatter Plot
```
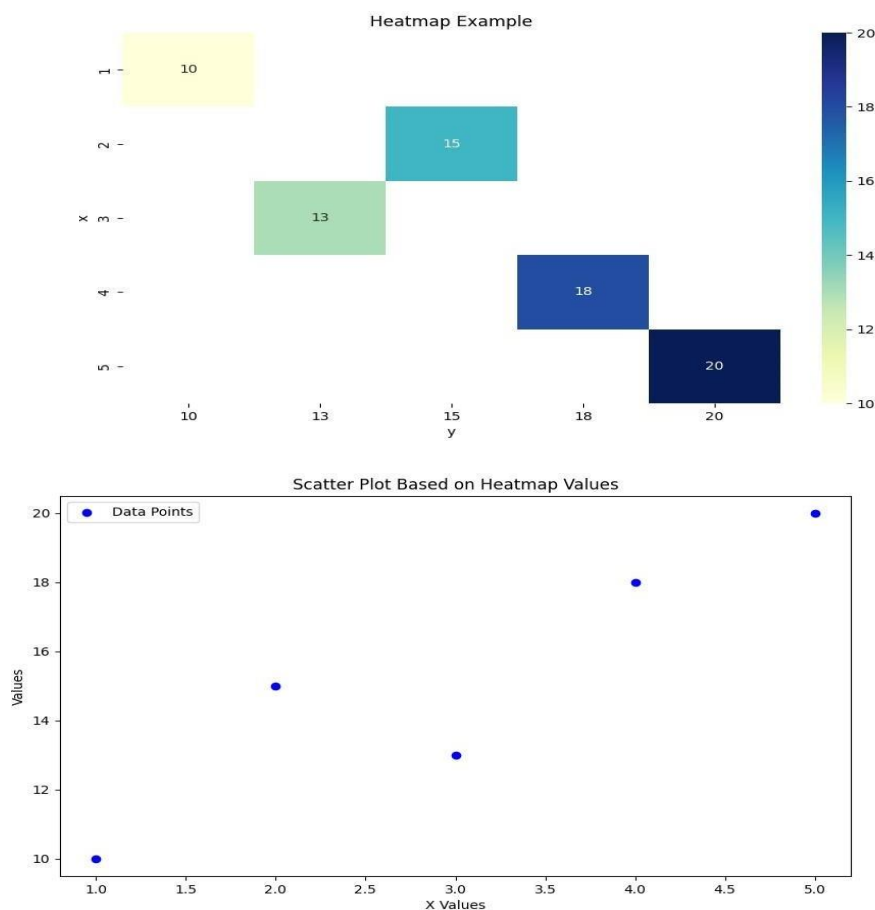
```
plt.figure(figsize=(10, 6))

plt.scatter(df['x'], df['value'], marker='o', color='blue', label='DataPoints')

plt.xlabel('X Values')

plt.ylabel('Values')

plt.title('Scatter Plot Based on Heatmap Values')

plt.legend()

plt.savefig('scatter_plot_example.png')  # Save to file

plt.close()
```

OUTPUT

RESULT

The program has been executed successfully and output is obtained

PROGRAM - 11

AIM

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.
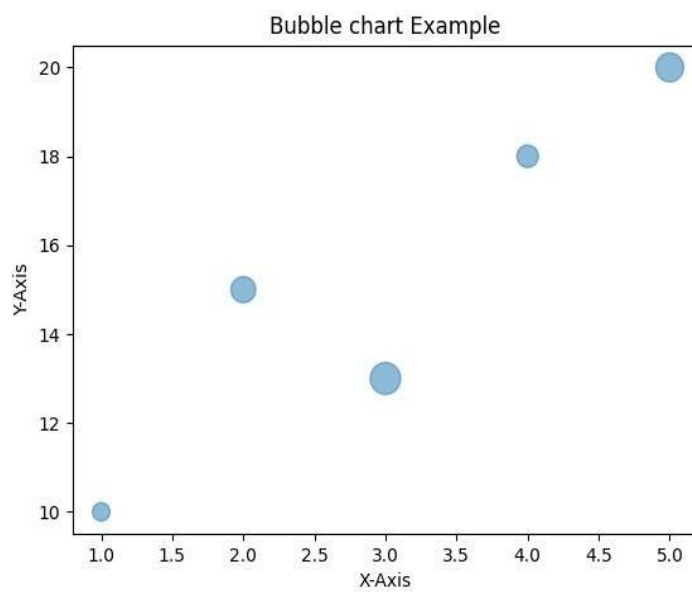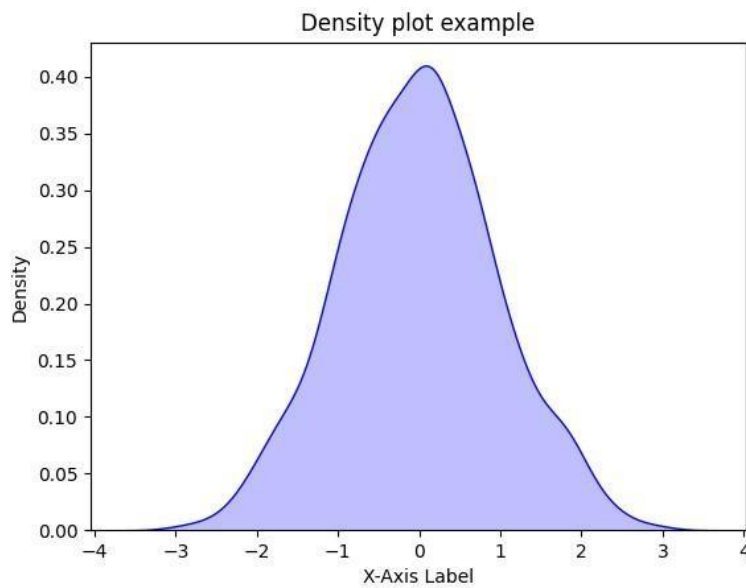
SOURCE CODE

//DensityChart.py

```python
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
data = np.random.randn(1000)
sns.kdeplot(data,fill=True,color='blue',label='Density Plot')
plt.xlabel('X-Axis Label')
plt.ylabel('Density')
plt.title('Density plot example')
plt.savefig('Density_plot.png')
```

```python
//BubbleDiagram.py import
matplotlib.pyplot as plt
x =[1,2,3,4,5]
y =[10,15,13,18,20]
sizes=[100,200,300,150,250]
plt.scatter(x,y,s=sizes,alpha=0.5)
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.title('Bubble chart Example')
plt.savefig('bubblechart.png')
```

OUTPUT

Density plot example

Bubble chart Example

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 12

AIM

Program to implement K-Nearest Neighbour Classification and find the accuracy of algorithm.

SOURCE CODE

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics importaccuracy_score,classification_report
data = pd.read_csv("iris.csv")
X = data.drop("species",axis=1) y = data["species"]
X_train, X_test, y_train,y_test = train_test_split(X,y, test_size=0.3,random_state=42)
knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X_train,y_train)
y_pred = knn_classifier.predict(X_test)
accuracy = accuracy_score(y_test,y_pred)
print("Accuracy:",accuracy)
report=classification_report(y_test,y_pred,target_names=data["species"].unique())
print("Classification Report:\n",report)
```

OUTPUT

```
/home/cep/PycharmProjects/pythonProject/.venv/bin/python /home/cep/Desktop/pythonProject/Knearest.py
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        19
  Versicolor       1.00      1.00      1.00        13
   Virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45


Process finished with exit code 0
```

RESULT

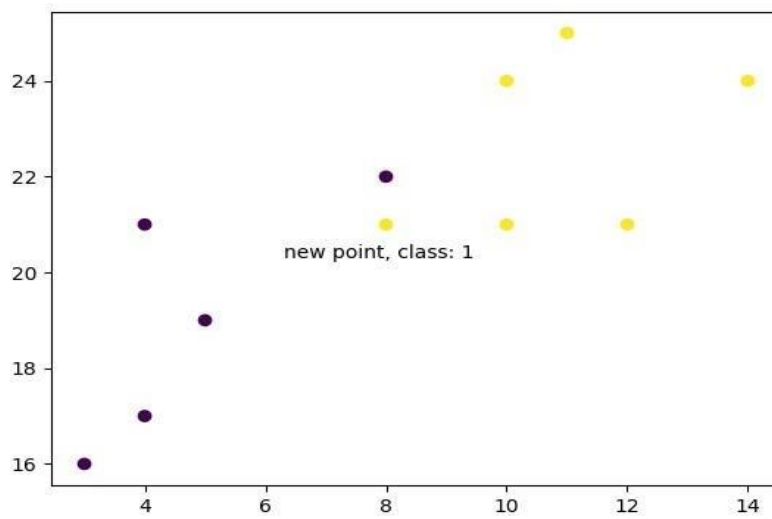The program has been executed successfully and output is obtained.

PROGRAM - 13

AIM

Program to demonstrate how the K-Nearest Neighbors (KNN) algorithm determines the class of a new data point.

SOURCE CODE

```python
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
data = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(data, classes) new_x = 8 new_y = 21
new_point = [(new_x, new_y)]
prediction = knn.predict(new_point)
plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.savefig("knn.png")
```

OUTPUT

new point, class: 1

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 14

AIM

Program to implement Naive Bayes Classifier.

SOURCE CODE

```
from sklearn .datasets
importload_iris
iris=load_iris()
X=iris.data
y=iris.target
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.4,random_state=)
from sklearn.naive_bayes import GaussianNB
gnb= GaussianNB()
gnb.fit(X_train,y_train)
y_pred=gnb.predict(X_test)
from sklearn import metrics
print("GaussNaive bayes model accuracy(in%):",metrics.accuracy_score(y_test,y_pred)*100)
```

OUTPUT

```
/home/cep/PycharmProjects/pythonProject/.venv/bin/python /home/cep/Desktop/pythonProject/naivebayes.py
GaussNaive bayes model accuracy(in %): 95.0

Process finished with exit code 0
```

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 15

AIM

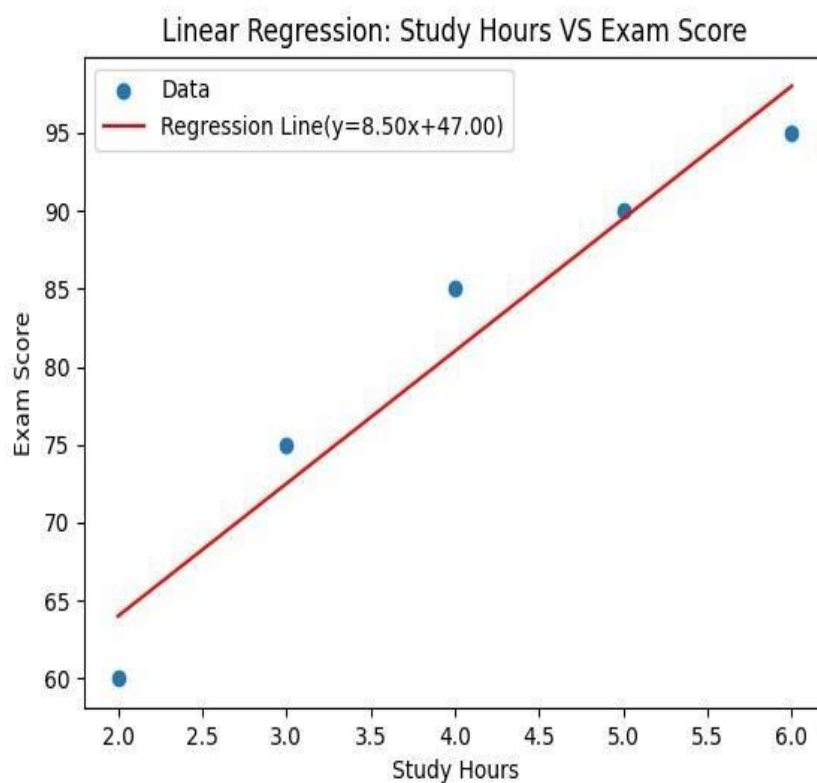Program to implement Simple Linear Regression and find r2 score.

SOURCE CODE

```
import numpy as np
from fontTools.subset import intersect
from sklearn.feature_selection import f_regression
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
X=np.array([2,3,4,5,6]).reshape(-1,1)
Y=np.array([60,75,85,90,95])
model=LinearRegression()
model.fit(X,Y)
Y_pred=model.predict(X)
slope=model.coef_[0]
intercept=model.intercept_
plt.scatter(X,Y,label='Data')
plt.plot(X,Y_pred,color='red',label=f'Regression
Line(y={slope:.2f}x+{intercept:.2f})')
plt.xlabel('Study Hours')
plt.ylabel('Exam Score')
plt.legend()
plt.title('Linear Regression: Study Hours VS Exam Score')
plt.savefig('Simple_Linear_Regression.png')
new_SH=int(input("Enter the number of hours:"))
pred_Sc=model.predict(np.array([[new_SH]]))
print(f'Predicted Exam Score for{new_SH}studyhours:{pred_Sc[0]:.2f}')
```

OUTPUT

```
/home/cep/PycharmProjects/pythonProject/.venv/bin/python /home/cep/Desktop/pythonProject/simpleLR.py
Enter the number of hours:8
Predicted Exam Score for 8 study hours: 115.00

Process finished with exit code 0
```


Linear Regression: Study Hours VS Exam Score

RESULT

The program has been executed successfully and output is obtained.
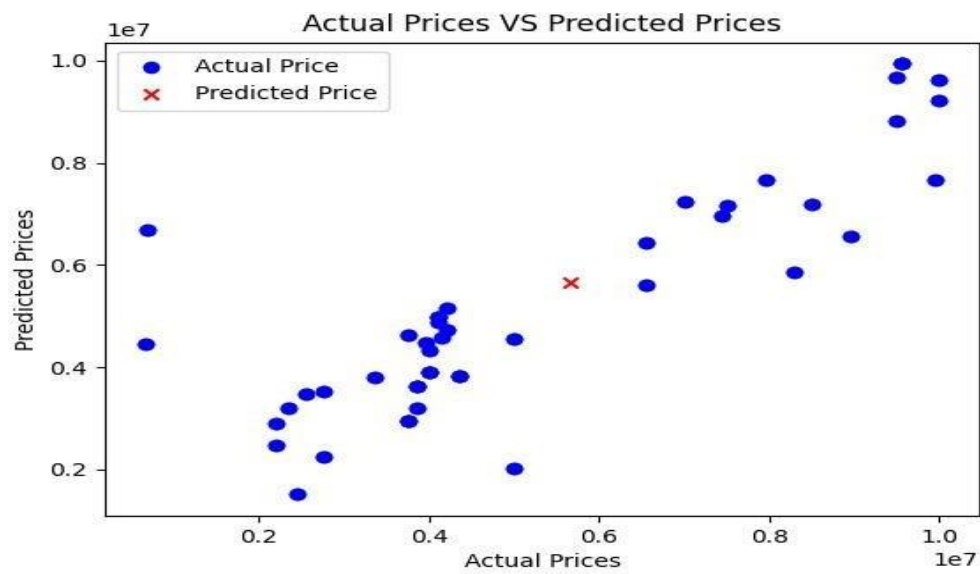
PROGRAM - 16

AIM

Program to implement Multiple Linear Regression and evaluate its performance.

SOURCE CODE

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
data = pd.read_csv("house-price.csv")
X = data[['No_Rooms','Sq_Foot','Age']].values
Y = data['Price'].values model = LinearRegression()
model.fit(X,Y)
Y_pred=model.predict(X)
plt.scatter(Y,Y_pred,color='blue')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual Prices VS Predicted Prices')
input_values=[int(input("Enterthenumberofrooms:")),
int(input("Enterthesquarefootage), int(input("Enter the Age :"))]
predicted_price=model.predict([input_values])
print("Predicted Price: ",predicted_price[0])
plt.scatter(Y,Y_pred,color='blue',label='Actual Price')
plt.scatter(predicted_price,predicted_price,color='red',marker='x',label='PredictedPrie)
plt.legend()
plt.savefig("MultipleLR.png")
```

OUTPUT

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 17

AIM

Program to implement Decision Tree Classification.

SOURCE CODE

```
import sys
import matplotlib
matplotlib.use('Agg')
import pandas as pd
from sklearn import DecisionTreeClassifier,plot_tree
import matplotlib.pyplot as plt

#Load the dataset
df=pd.read_csv('dec_csv')
# Print initial DataFrameinfo
print("Initial DataFrame:")
print(df.info())
print(df.head())

# Map categorical variables to numerical values
nationality_map = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] =
df['Nationality'].map(nationality_map)
d = {'YES': 1, 'NO': 0} df['Go'] = df['Go'].map(d)

# Check for NaN values in the target variable
print("Checking for NaN values in 'Go' column...")
print(df['Go'].isnull().sum())

if df['Go'].isnull().any():
    print("NaN values found in 'Go' column. Dropping rows with NaN values.")
df = df.dropna(subset=['Go'])
```

```
print("DataFrame after dropping NaN values:")
print(df.info())
print(df.head())

# Check if there's enough data
if df.shape[0] == 0:
    print("No data available for training. Please check your dataset.")
sys.exit(1)

# Define features and target variable
features=['Age','Experience','Rank','Nationality']
X = df[features]
 y = df['Go']
dtree=DecisionTreeClassifier()
dtree.fit(X, y)

# Plot the decision tree plt.figure(figsize=(10, 8))
plot_tree(dtree,feature_names=features,filled=Tru)
plt.title('Decision Tree Visualization')
plt.savefig('decision_tree.png')
plt.close()
sys.stdout.flush()

//decision_data.csv
"Age","Experience","Rank","Nationality","Go"
36,10,9,UK,NO
42,12,4,USA,NO
23,4,6,N,NO
52,4,4,USA,NO
43,21,8,USA,YES
44,14,5,UK,NO
66,3,7,N,YES
35,14,9,UK,YES
52,13,7,N,YES
```
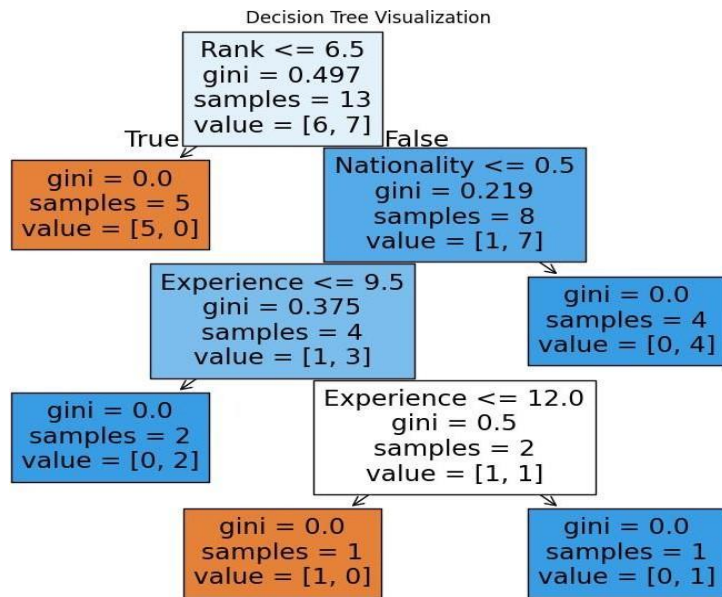
35,5,9,N,YES

24,3,5,USA,NO

18,3,7,UK,YES

45,9,9,UK,YES

OUTPUT

Decision Tree Visualization

RESULT

The program has been executed successfully and output is obtained.

PROGRAM – 18

AIM

Program to implement K-Means Classification.

SOURCE CODE

```python
import numpy as np
import matplotlib.pyplot as plt
data=np.array([[2, 3], [3, 4], [5, 6], [9, 10], [10,8],[12,12]])
k=2
  centroids=data[np.random.choice(data.shape[0],k,replace=
  False)]
num_iterations=100
for iteration in range(num_iterations):
distances=np.linalg.norm(data[:, np.newaxis] - centroids,axis=2)
labels = np.argmin(distances, axis=1)
print(labels)
plt.scatter(data[:, 0], data[:, 1], c=labels)
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='X', s=200)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title(f"Iteration {iteration}")
plt.show()
new_centroids = np.array([data[labels == i].mean(axis=0)
for i in range(k)])
if np.array_equal(centroids, new_centroids):
break
centroids = new_centroids
print(distances)
print("Final Cluster Assignments:", labels)
```

print("Final Cluster Centroids:", centroids)

OUTPUT

```
[[10.88321848   1.88561808]
 [ 9.4751136    0.47140452]
 [ 6.66666667   2.3570226 ]
 [ 1.33333333   8.01387685]
 [ 2.02758751   7.60847481]
 [ 2.60341656  11.57103664]]
Final Cluster Assignments: [1 1 1 0 0 0]
Final Cluster Centroids: [[10.33333333 10.          ]
 [ 3.33333333   4.33333333]]
```
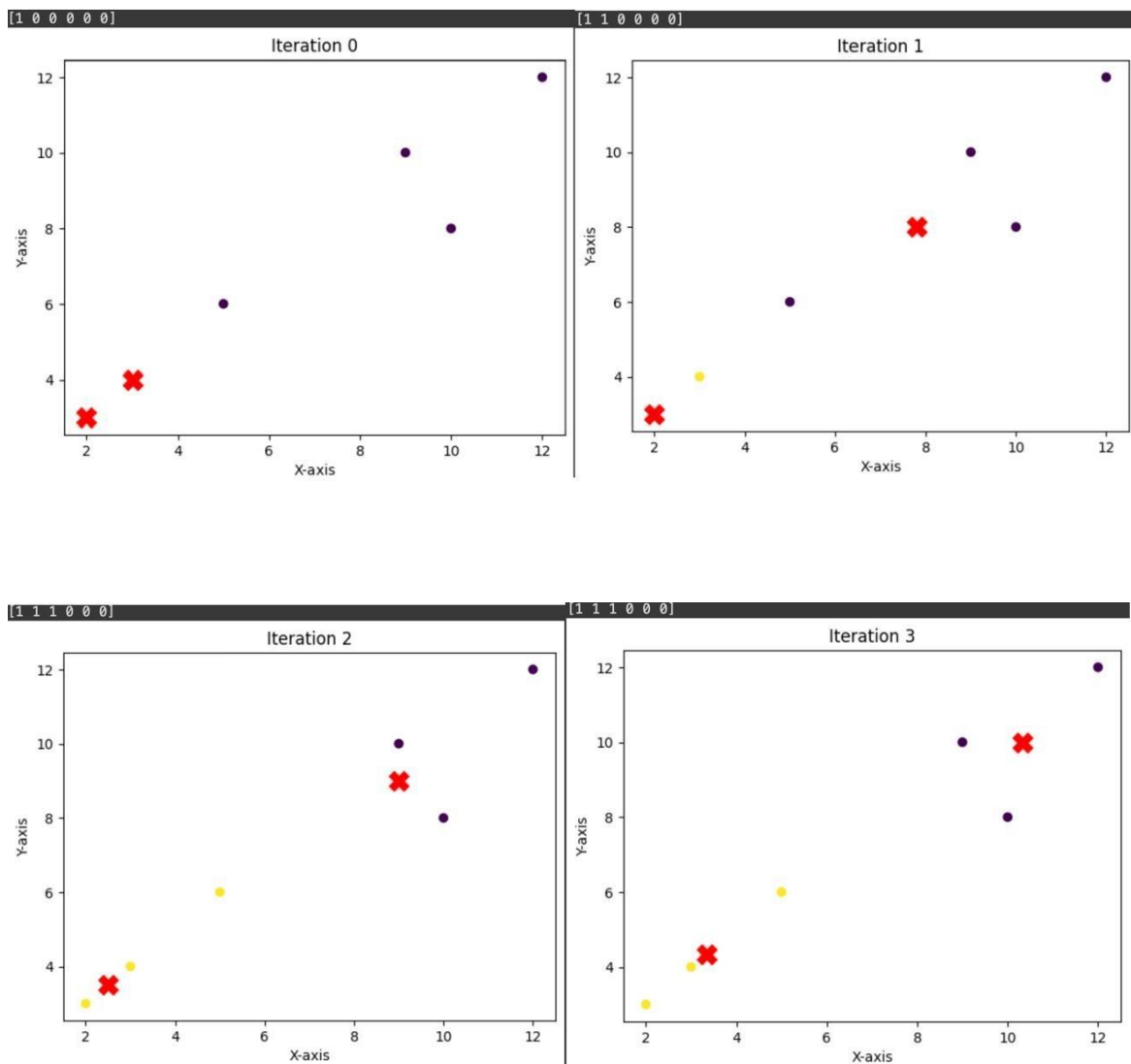

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 19

AIM

Program to implement Part-Of-Speech (POST) Tagging.

SOURCE CODE

import nltk from nltk

import pos_tag

from nltk.tokenize import word_tokenize

sentence = "part-of-speech tagging is important for natural language processing"

words = word_tokenize(sentence)

pos_tags = pos_tag(words)

print (pos_tags)

OUTPUT

```
/home/cep/PycharmProjects/pythonProject/.venv/bin/python /home/cep/Desktop/pythonProject/p15_POSTtagging.py
[('part-of-speech', 'JJ'), ('tagging', 'NN'), ('is', 'VBZ'), ('important', 'JJ'), ('for', 'IN'), ('natural', 'JJ'), ('language', 'NN'), ('processing', 'NN')]

Process finished with exit code 0
```

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 20

AIM

Program to implement N-Gram Modeling.

SOURCE CODE

```
from nltk import bigrams,word_tokenize
sentence="Natural languageprocessingisfascinating."
words = word_tokenize(sentence)
bigrams_list = list(bigrams(words))
print (bigrams_list)
```

OUTPUT

```
/home/cep/PycharmProjects/pythonProject/.venv/bin/python /home/cep/Desktop/pythonProject/P16_NGram.py
[('Natural', 'language'), ('language', 'processing'), ('processing', 'is'), ('is', 'fascinating'), ('fascinating', '.')]

Process finished with exit code 0
```

RESULT

The program has been executed successfully and output is obtained.

PROGRAM – 21

AIM

Program to implement a Feed Forward Network using Iris Data Set.

SOURCE CODE

```
from scipy.signal import impulse
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.datasets import load_iris
from tensorflow.keras import models,layers
from Knearest import X_train,X_test,y_train,y_test
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)
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
label_encoder=LabelEncoder()
y_train=label_encoder.fit_transform(train)
y_test=label_encoder.transform(y_test)
model= models.Sequential()
model.add(layers.Dense(64,activation='relu',input_shape=(4,)))
model.add(layers.Dense(32,activation='relu'))
model.add(layers.Dense(3,activation='softmax'))
model.compile(optimizer='adam',loss='sparse_categorical_cross entropy'acy'])
model.fit(X_train,y_train,epochs=50,batch_size=8,validation_split(0.1)
test_loss,test_acc=model.evaluate(X_test,y_test)
print(f'Test accuracy: {test_acc}')
```

OUTPUT

```
14/14 ──────────── 0s 2ms/step - accuracy: 0.9645 - loss: 0.0613 - val_accuracy: 0.9167 - val_loss: 0.3557
Epoch 37/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9922 - loss: 0.0458 - val_accuracy: 0.9167 - val_loss: 0.3319
Epoch 38/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9836 - loss: 0.0638 - val_accuracy: 0.9167 - val_loss: 0.3516
Epoch 39/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9588 - loss: 0.0805 - val_accuracy: 0.9167 - val_loss: 0.3342
Epoch 40/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9605 - loss: 0.0626 - val_accuracy: 0.9167 - val_loss: 0.3325
Epoch 41/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9561 - loss: 0.0700 - val_accuracy: 0.9167 - val_loss: 0.3370
Epoch 42/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9875 - loss: 0.0676 - val_accuracy: 0.9167 - val_loss: 0.3182
Epoch 43/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9814 - loss: 0.0536 - val_accuracy: 0.9167 - val_loss: 0.3574
Epoch 44/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9719 - loss: 0.0699 - val_accuracy: 0.9167 - val_loss: 0.3827
Epoch 45/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9796 - loss: 0.0570 - val_accuracy: 0.9167 - val_loss: 0.3512
Epoch 46/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9948 - loss: 0.0439 - val_accuracy: 0.9167 - val_loss: 0.3713
Epoch 47/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9838 - loss: 0.0576 - val_accuracy: 0.9167 - val_loss: 0.3596
Epoch 48/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9683 - loss: 0.0713 - val_accuracy: 0.9167 - val_loss: 0.3529
Epoch 49/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9788 - loss: 0.0555 - val_accuracy: 0.9167 - val_loss: 0.3783
Epoch 50/50
14/14 ──────────── 0s 2ms/step - accuracy: 0.9901 - loss: 0.0508 - val_accuracy: 0.9167 - val_loss: 0.3655
1/1 ──────────── 0s 13ms/step - accuracy: 1.0000 - loss: 0.0399
Test accuracy: 1.0
```

RESULT

The program has been executed successfully and output is obtained.

PROGRAM - 22

AIM

For given text
- Perform word
- Sentence tokenization
- Remove the stop words from the given text
- Create n-grams

SOURCE CODE

```
import nltk
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')
from nltk import ngrams
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize,word_tokenize
print("PJR24MCA-2005:ANJIMA C P ")
print("BATCH: MCA 2024-2026")
text1="The data given satisfies the requirements for model generation.This is used in Data Science Lab"
print("Sentence Tokenization")
print(sent_tokenize(text1))
print("Word Tokenization:")
print(word_tokenize(text1))
text=word_tokenize(text1)
text2=[word for word in text if word not in stopwords.words('english')]
print("Removing stop_words:")
print(text2)
print("")
print("n grams")
unigrams=ngrams(text2,3)
```

for grams in unigrams:

   print(grams)

OUTPUT

```
/Users/anjimacp/PycharmProjects/PythonProject5/.venv1/bin/python /Users/anjimacp/PycharmProjects/PythonProject5/ngrams.py
PJR24MCA-2005:ANJIMA C P
BATCH: MCA 2024-2026
Sentence Tokenization
['The data given satisfies the requirements for model generation.This is used in Data Science Lab']
Word Tokenization:
['The', 'data', 'given', 'satisfies', 'the', 'requirements', 'for', 'model', 'generation.This', 'is', 'used', 'in', 'Data', 'Science', 'Lab']
Removing stop_words:
['The', 'data', 'given', 'satisfies', 'requirements', 'model', 'generation.This', 'used', 'Data', 'Science', 'Lab']

n grams
('The', 'data', 'given')
('data', 'given', 'satisfies')
('given', 'satisfies', 'requirements')
('satisfies', 'requirements', 'model')
('requirements', 'model', 'generation.This')
('model', 'generation.This', 'used')
('generation.This', 'used', 'Data')
('used', 'Data', 'Science')
('Data', 'Science', 'Lab')
```

RESULT

The program has been executed successfully and output is obtained.