20MCA241 – Data Science Lab

Lab Report Submitted By

SIDDHARTH DILEEP

AJC22MCA-2082

In Partial Fulfilment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS
(MCA TWO YEAR)
[Accredited by NBA]

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2022-2024

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the lab report, "**20MCA241 DATA SCIENCE LAB"** is the bonafide work of **SIDDHARTH DILEEP (AJC22MCA-2082)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year **2023-24.**

Ms. Ankitha Philip   Rev. Fr. Dr. Rubin Thottupurathu Jose

Lab In- Charge   **Head of the Department**

Internal Examiner   **External Examiner**

Course Code	Course Name	Syllabus Year	L-T-P-C
20MCA241	Data Science Lab	2020	0-1-3-2

VISION

To promote an academic and research environment conducive for innovation centric technical education.

MISSION

MS1 - Provide foundations and advanced technical education in both theoretical and applied Computer Applications in-line with Industry demands.

MS2 - Create highly skilled computer professionals capable of designing and innovating real life solutions.

MS3 - Sustain an academic environment conducive to research and teaching focused to generate up-skilled professionals with ethical values.

MS4 - Promote entrepreneurial initiatives and innovations capable of bridging and contributing with sustainable, socially relevant technology solutions.

COURSE OUTCOME

CO	Outcome	Target
CO1	Use different python packages to perform numerical calculations, statistical computations and data visualization.	60.2
CO2	Use different packages and frameworks to implement regression and classification algorithms.	60.2
CO3	Use different packages and frameworks to implement text classification using SVM and clustering using K-means.	60.2
CO4	Implement convolutional neural network algorithm using Keras framework.	60.2
CO5	Implement programs for web data mining and natural language processing using NLTK.	60.2

COURSE END SURVEY

CO	Survey Question	Answer Format
CO1	To what extend you are able to use different python packages to perform numerical calculations, statistical computations and data visualization?	Excellent/Very Good/Good/Satisfactory/Poor
CO2	To what extend you are able to use different packages and frameworks to implement regression and classification algorithms?	Excellent/Very Good/Good/Satisfactory/Poor
CO3	To what extend you are able to use different packages and frameworks to implement text classification using SVM and clustering using K-means?	Excellent/Very Good/Good/Satisfactory/Poor
CO4	To what extend you are able to implement convolutional neural network algorithm using Keras framework?	Excellent/Very Good/Good/Satisfactory/Poor
CO5	To what extend you are able to implement programs for web data mining and natural language processing using NLTK?	Excellent/Very Good/Good/Satisfactory/Poor

CONTENT

Experiment No.: 1

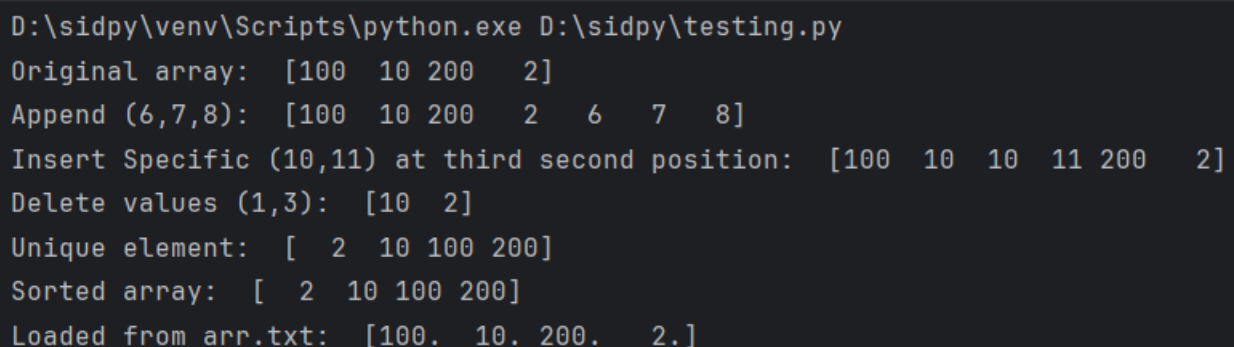**Aim:** Create a numpy array and perform the following operations

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

CO1: Use different python packages to perform numerical calculations, statistical computations and data   visualization.

Procedure

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

Output Screenshot

```
D:\sidpy\venv\Scripts\python.exe D:\sidpy\testing.py
Original array:  [100  10 200   2]
Append (6,7,8):  [100  10 200   2   6   7   8]
Insert Specific (10,11) at third second position:  [100  10  10  11 200   2]
Delete values (1,3):  [10   2]
Unique element:  [  2  10 100 200]
Sorted array:  [  2  10 100 200]
Loaded from arr.txt:  [100.  10. 200.   2.]
```

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

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:
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.

CO1: Use different python packages to perform numerical calculations, statistical computations and data   visualization.

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
print ("sum is",result_add)
result_mul=arr1 * arr2
print ("Product is",result_mul)
meanarr1=np.mean(result_add)
print("mean is",meanarr1)
maxarr=max(result_mul)
print("maximum is",maxarr)
```
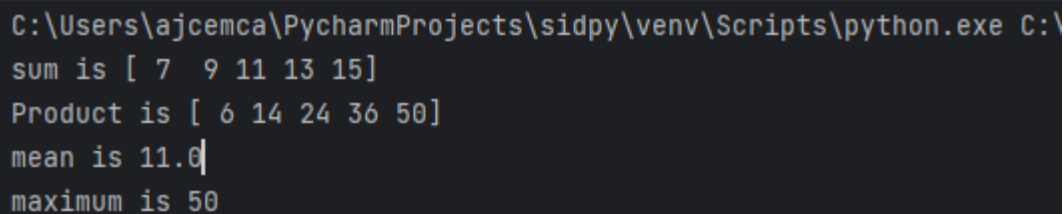
Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\sidpy\venv\Scripts\python.exe C:\
sum is [ 7  9 11 13 15]
Product is [ 6 14 24 36 50]
mean is 11.0
maximum is 50
```

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

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])
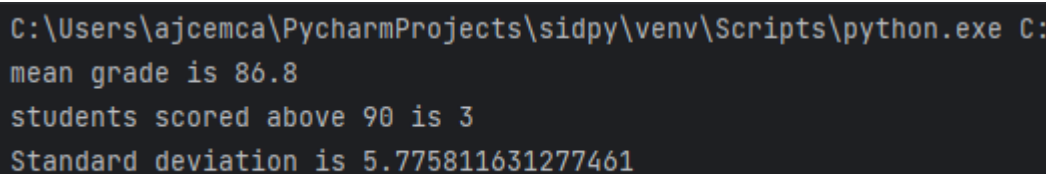Write NumPy code to answer the following questions:

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**.

CO1: Use different python packages to perform numerical calculations, statistical computations and data   visualization.

Procedure

```
import numpy as np
grades = np.array([85, 90, 78, 92, 88, 76, 95, 89, 84, 91])
meangr=np.mean(grades)
print("mean grade is",meangr)
greatnine=grades[grades>90]
stud=len(greatnine)
print("students scored above 90 is",stud)
standarddevia=np.std(grades)
print("Standard deviation is",standarddevia)
```

Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\sidpy\venv\Scripts\python.exe C:
mean grade is 86.8
students scored above 90 is 3
Standard deviation is 5.775811631277461
```

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

Experiment No.: 4

Aim: Matrix Operations with NumPy
>   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:
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.
5. Calculate the determinant of matrix_B and store it in determinant_B.
6. Find the eigenvalues and eigenvectors of matrix_A and store them in eigenvalues_A and eigenvectors_A.
7. Find SVD of a matrix

 Print the results of each operation.

CO1: Use different python packages to perform numerical calculations, statistical computations and data   visualization.

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 = np.transpose(matrix_A)
determinant_B = np.linalg.det(matrix_B)
eigenvalues_A, eigenvectors_A = np.linalg.eig(matrix_A)
U, S, VT = np.linalg.svd(matrix_A) #svd

print("Matrix Sum:",matrix_sum)
print("Matrix Element-wise Product:",matrix_product)
print("Matrix Dot Product:",matrix_dot)
print("Matrix A Transpose:",matrix_A_transpose)
print("Determinant of Matrix B:",determinant_B)
print("Eigenvalues of Matrix A:",eigenvalues_A)
print("Eigenvectors of Matrix A:",eigenvectors_A)
print("U matrix:", U)
print("S matrix (singular values):", S)
print("VT matrix (transpose of V):", VT)

Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\sidpy\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\sidpy\ml
Matrix Sum: [[10 10 10]
 [10 10 10]
 [10 10 10]]
Matrix Element-wise Product: [[ 9 16 21]
 [24 25 24]
 [21 16  9]]
Matrix Dot Product: [[ 30  24  18]
 [ 84  69  54]
 [138 114  90]]
Matrix A Transpose: [[1 4 7]
 [2 5 8]
 [3 6 9]]
Determinant of Matrix B: 0.0
Eigenvalues of Matrix A: [ 1.61168440e+01 -1.11684397e+00 -1.30367773e-15]
Eigenvectors of Matrix A: [[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735   0.61232756  0.40824829]]
U matrix: [[-0.21483724  0.88723069  0.40824829]
 [-0.52058739  0.24964395 -0.81649658]
 [-0.82633754 -0.38794278  0.40824829]]
S matrix (singular values): [1.68481034e+01 1.06836951e+00 4.41842475e-16]
VT matrix (transpose of V): [[-0.47967118 -0.57236779 -0.66506441]
```

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

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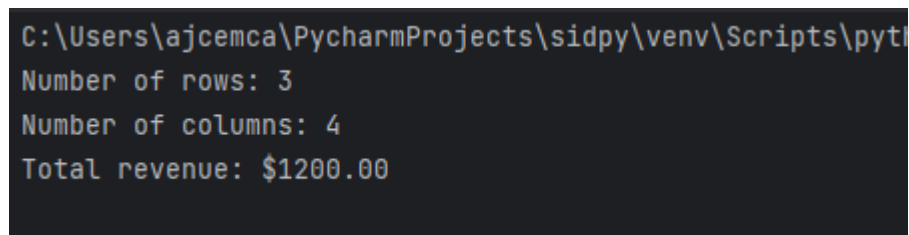2. What is the total revenue for all the sales?

CO1: Use different python packages to perform numerical calculations, statistical computations and data visualization.

Procedure

```
import pandas as pd
df = pd.read_csv("salesdata.csv")
num_rows, num_columns = df.shape
total_revenue = df["revenue"].sum()

print(f"Number of rows: {num_rows}")
print(f"Number of columns: {num_columns}")
print(f"Total revenue: ${total_revenue:.2f}")
```

Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\sidpy\venv\Scripts\pyt
Number of rows: 3
Number of columns: 4
Total revenue: $1200.00
```

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

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:

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.

CO1: Use different python packages to perform numerical calculations, statistical computations and data   visualization.
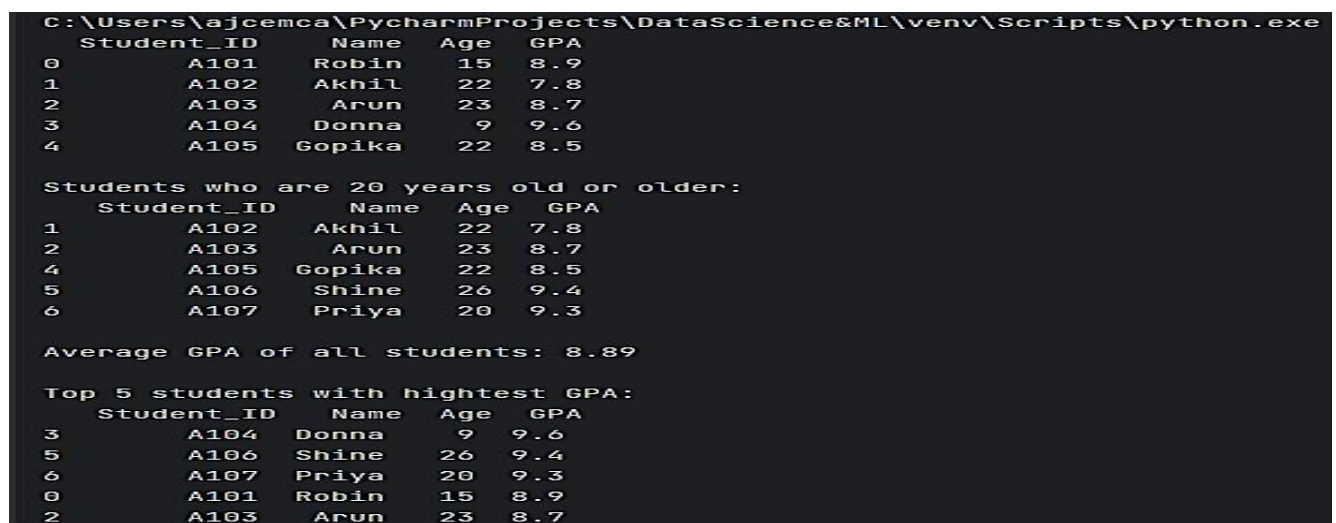
Procedure

```
import pandas as pd
df = pd.read_csv("student_data.csv")
print(df.head(), "\n")
age = df[df['Age'] > 19]
print("Students who are 20 years old or older: \n", age)

print("\nAverage GPA of all students:",df['GPA'].mean().round(2))

data1 = df.sort_values(by='GPA', ascending= False)
print("\nTop 5 students with hightest GPA: \n", data1.head(5))

data2 = df.groupby('Age')['GPA'].mean().reset_index()
print("\nAverage GPA by age group: \n", data2)
```

Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\DataScience&ML\venv\Scripts\python.exe
    Student_ID      Name   Age   GPA
0         A101     Robin    15   8.9
1         A102     Akhil    22   7.8
2         A103      Arun    23   8.7
3         A104     Donna     9   9.6
4         A105    Gopika    22   8.5

Students who are 20 years old or older:
    Student_ID      Name   Age   GPA
1         A102     Akhil    22   7.8
2         A103      Arun    23   8.7
4         A105    Gopika    22   8.5
5         A106     Shine    26   9.4
6         A107     Priya    20   9.3

Average GPA of all students: 8.89

Top 5 students with hightest GPA:
    Student_ID      Name   Age   GPA
3         A104     Donna     9   9.6
5         A106     Shine    26   9.4
6         A107     Priya    20   9.3
0         A101     Robin    15   8.9
2         A103      Arun    23   8.7
```

```
Average GPA by age group:
    Age   GPA
0     9  9.60
1    15  8.90
2    20  9.30
3    22  8.15
4    23  8.70
5    26  9.40
```

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

Experiment No.: 7
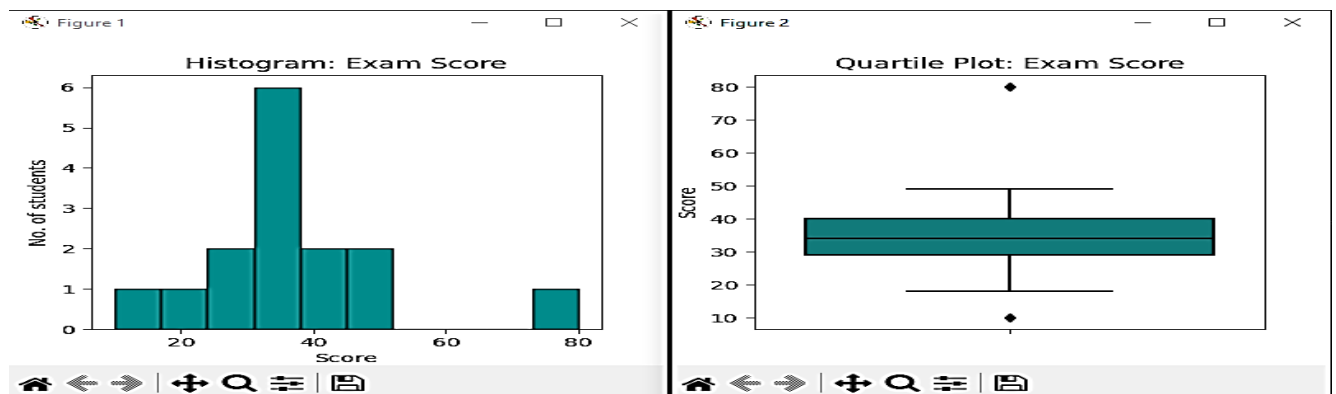
Aim: Histogram and Quartile Plot:
- a. **Question**: Use a dataset of your choice (e.g., exam scores of students, employee salaries, or any other numerical data). Create a histogram to visualize the data's distribution. Afterward, plot quartiles (e.g., Q1, Q2, Q3) on the same graph. Answer the following questions:
  - i.   What does the histogram reveal about the data's distribution?
  - ii. How do the quartiles relate to the histogram?
  - iii. Are there any outliers in the data, and if so, how do they affect the quartiles?
- b. **Output:** Provide the histogram and quartile plot along with a written analysis.

CO1: Use different python packages to perform numerical calculations, statistical computations and data   visualization.

Procedure

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
marks = np.array([10,18,34,
        37, 33, 38,
        34, 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.figure(figsize=(4, 4))
sns.boxplot(y=marks, color='darkcyan')
plt.title('Quartile Plot: Exam Score')
plt.ylabel('Score')
plt.show()
```

Output Screenshot

a. The histogram reveals that the largest distribution of score is between 30 and 38 with 6 students. The distribution between 10-24 and 73-80 is the same with 1 student.

b. The Quartile plot shows that the distribution in lower quartile is less than the upper quartile which reveals a higher score among the students.The InterQuartile range(IQR) between 28 and 40 reveals where the majority scores are distributed. We also obtain a median of 34 for the data.

c. It is revealed that two outliers at 10 and 80 exist which are translated better in Quartile plot than histogram. These outliers remain clearly out of the minimum and maximum of the plot thereby signifying an unusual difference from the usual distribution.

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

Experiment No.: 8

Aim: Distribution Chart and Scatter Plot:
a. **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:
   i. What do the distribution charts reveal about each variable?
   ii. Is there a correlation between the two variables based on the scatter plot?
   iii. Can you identify any patterns or trends in the data?
Output: Present the distribution charts, scatter plot, and your observations in a report.

CO1: Use different python packages to perform numerical calculations, statistical computations and data   visualization.

Procedure

Heat Map charts:-
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

x_values = [1, 2, 3, 4, 5]
y_values = [10, 15, 13, 18, 20]
data_values = [10, 15, 13, 18, 20]
df = pd.DataFrame({'x': x_values, 'y': y_values, 'value': data_values})
heatmap_data = df.pivot_table(index='x', columns='y', values='value')
fig, ax = plt.subplots(figsize=(10, 6))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', cbar=True)
plt.title('Heatmap Example')
plt.show()
```
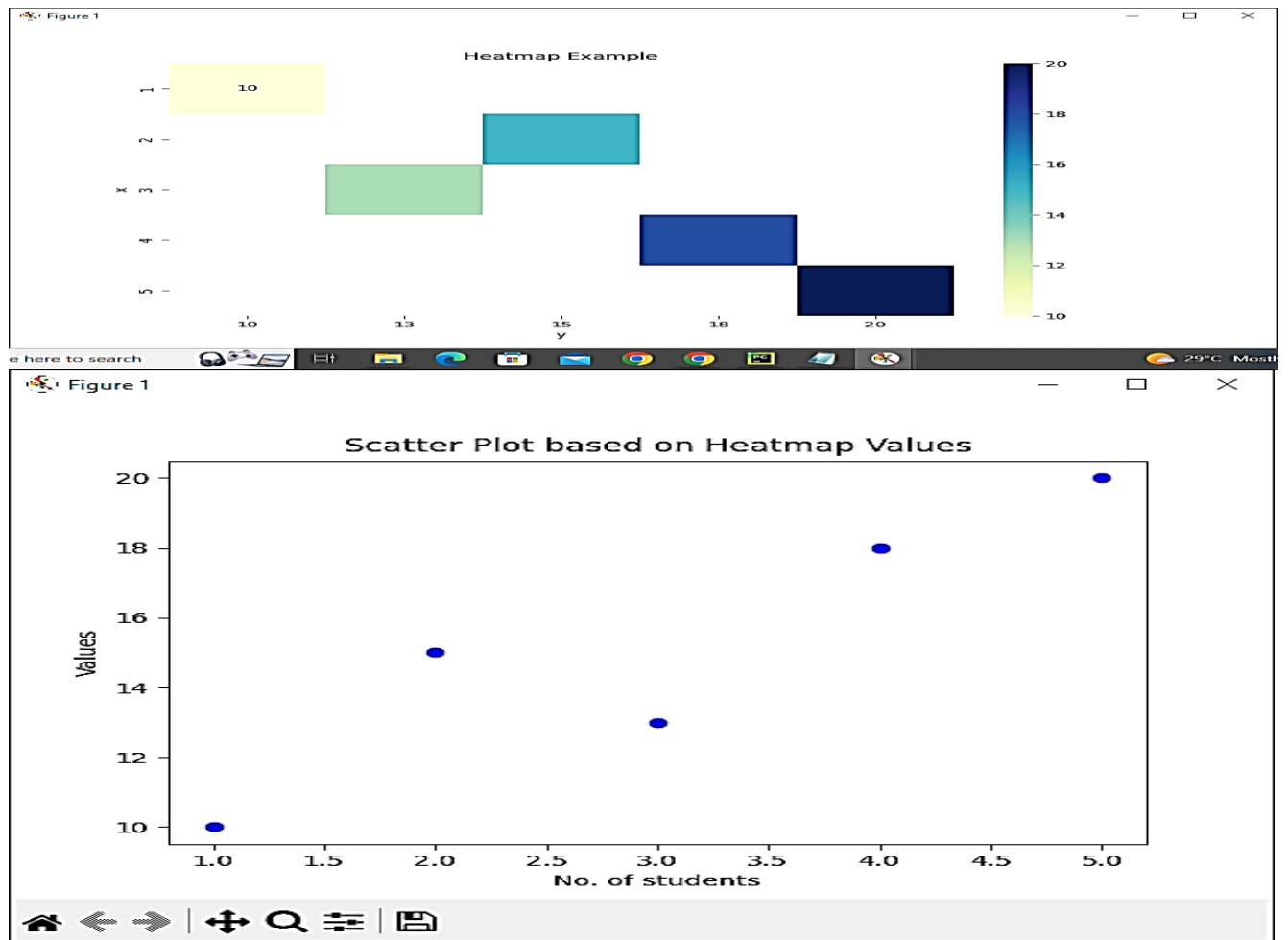
Scatter Plot:-

```
x_values = [1, 2, 3, 4, 5]
data_values = [10, 15, 13, 18, 20]

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

plt.scatter(df['x'], df['value'], marker='o', color='blue', label='Data Points')
plt.xlabel('No. of students')
plt.ylabel('Values')
plt.title('Scatter Plot based on Heatmap Values')
plt.show()
```

Output Screenshot

a.   In the heatmap, the values are symmetrically distributed along the x and y axes, forming a diagonal pattern from the bottom-left to the top-right. This indicates that there might be some correlation or relationship between 'x' and 'y' values.

In the scatter plot, there is a clear upward trend in the data points as 'x' increases. This suggests a positive correlation between 'x' and 'value'.   As the number of students (x-values) increases, the 'values' tend to increase as well.

The data points are relatively closely clustered around the trendline, indicating a strong correlation. There doesn't appear to be any outliers or significant deviations from the trendline.

b. Based on the scatter plot and the observed trend, it is evident that there is a positive correlation between the two variables ('x' and 'value'). As 'x' increases, 'value' tends to increase as well.

c. The primary pattern or trend identified in the data is a positive linear relationship. As the number of students (x-values) increases, the 'values' also increase, suggesting that there might be a positive impact of increasing the number of students on the 'values'. In summary, the distribution charts and scatter plot reveal that there is a positive correlation between the two variables ('x' and 'value'), and the data follows a clear upward trend. Increasing the number of students is associated with higher 'values'.

Result: The program was executed successfully and the output was obtained. Thus, CO1 has been attained.

Experiment No.: 9

Aim: Bubble Chart and Density Chart:
   a. **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:
      i.   How does the bubble chart help in visualizing multivariate data?
      ii.   What insights can you gain from the density chart in terms of data concentration?
      iii. Are there any interesting patterns or outliers in the data?

.

CO1: Use different python packages to perform numerical calculations, statistical computations and data   visualization.

Procedure

Density Chart:-
```
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.show()
```

Bubble Diagram
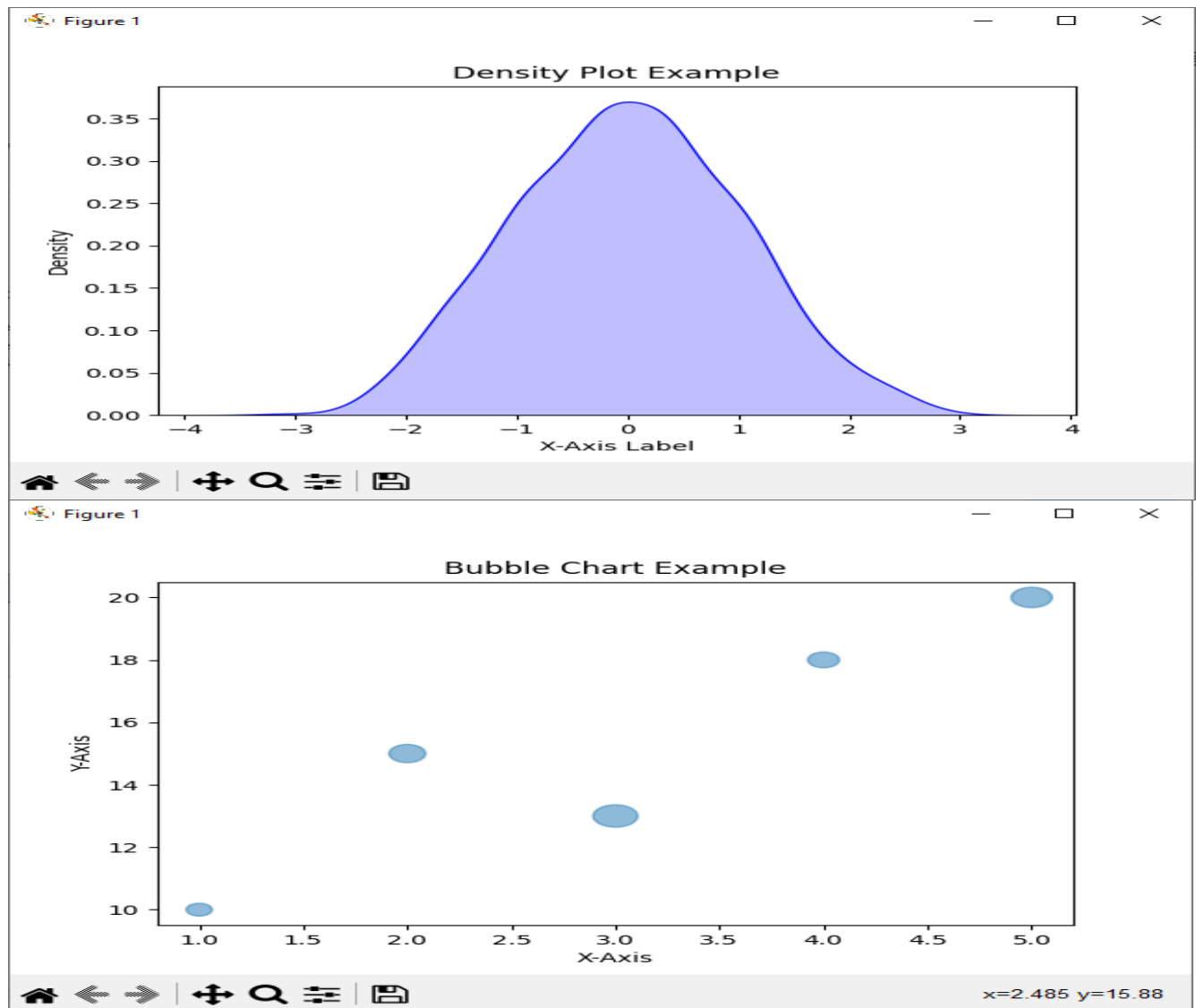```
import matplotlib.pyplot as plt

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.show()
```

## Output Screenshot



i. Bubble Chart for Multivariate Data Visualization:
A bubble chart is effective for visualizing multivariate data as it incorporates three variables in a two-dimensional space. In this case:
X-Axis: Represents one variable.
Y-Axis: Represents another variable.
Bubble Size: Represents a third variable.
The size of each bubble indicates the magnitude of the third variable. This allows you to quickly identify patterns and relationships between three different aspects of the data. For example, in the given code, you can observe how the size of the bubbles varies, providing insights into the relationship between 'x,' 'y,' and 'sizes.'

ii. Density Chart Insights:
The density chart (Kernel Density Estimate - KDE) is used to visualize the distribution of a univariate data set. In the provided code:

X-Axis: Represents the data points.
Y-Axis: Represents the estimated density of those data points.
Insights from the density chart:
Data Concentration: The peaks in the density plot indicate regions where the data is concentrated. In areas with higher density, you can infer that there are more data points.

iii. Patterns and Outliers:
Density Chart (KDE): The density chart may reveal peaks and troughs, suggesting where the data is concentrated or sparse. In this example, since the data is generated randomly, you may not observe specific patterns or outliers.
Bubble Chart: Patterns in the bubble chart can be observed by examining the distribution and size of bubbles. If there are clusters of large or small bubbles, it indicates patterns or trends in the multivariate data. In this specific example, the data is manually set, and no distinct patterns or outliers are apparent. It's important to note that for more meaningful insights, real-world data with specific patterns or characteristics is often required. The provided code uses random data, limiting the ability to make specific observations about patterns or outliers.

**Result:** The program was executed successfully and the output was obtained. Thus, CO1 has been attained.
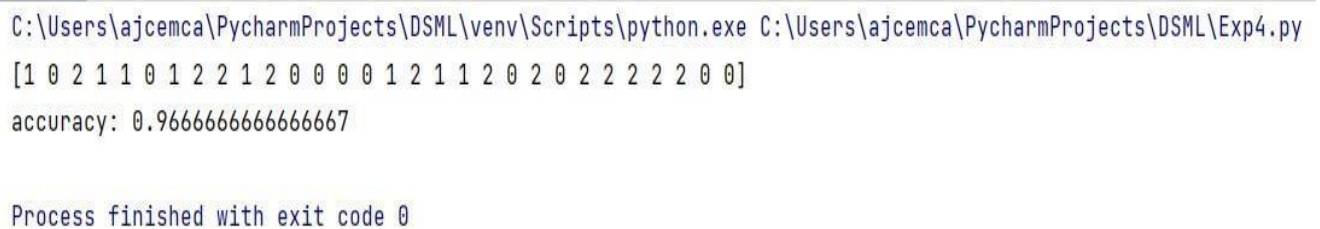
## Experiment No.: 10

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

**CO2:** Use different packages and frameworks to implement regression and classification algorithms.

## Procedure

```
from sklearn.neighbors import KNeighborsClassifier from
sklearn.model_selection import train_test_split from
sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
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)
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train,y_train)
print(knn.predict(x_test))
V=knn.predict(x_test)
result=accuracy_score(y_test,V)
print("Accuracy= ",result)
```

Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\DSML\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\DSML\Exp4.py
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
accuracy: 0.9666666666666667

Process finished with exit code 0
```

Result: The program was executed successfully and the output was obtained. Thus, CO2 has been attained.

Experiment No.: 11

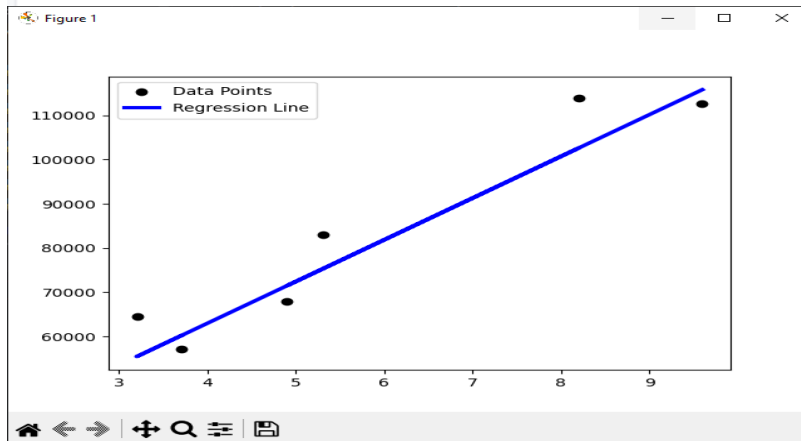**Aim:** Program to implement Simple Linear Regression and find r2 score.

CO2: Use different packages and frameworks to implement regression and classification algorithms.

Procedure

```
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 r2_score
data=pd.read_csv('Salary_Data.csv')
x=data['YearsExperience'].values.reshape(-1,1)
y=data['Salary'].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42) LR=LinearRegression()
LR.fit(x_train,y_train)
D=LR.predict(x_test)  r2 =
r2_score(y_test, D)
print("R2 Score: ", r2)
plt.scatter(x_test,y_test,color='black',label='Data Points')
plt.plot(x_test,D,color='blue', linewidth=3,label='Regression Line')
plt.xlabel='YearsExperience'
plt.ylabel='Salary'
plt.legend() plt.show()
```

Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python
[115790.21011287   71498.27809463 102596.86866063   75267.80422384
   55477.79204548   60189.69970699]
r2 score: 0.9024461774180497
```


Result: The program was executed successfully and the output was obtained. Thus, CO_2 has been attained.
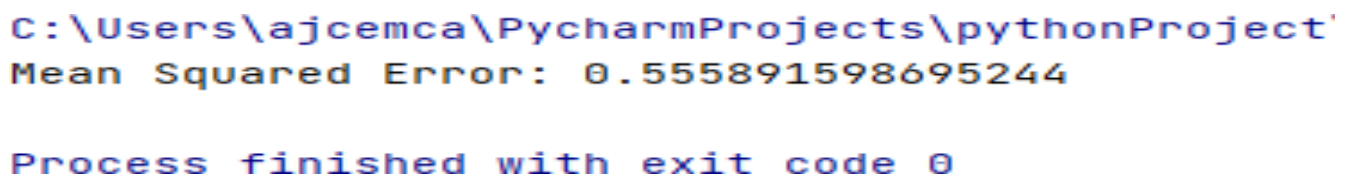
Experiment No.: 12

Aim: Program to implement Multiple Linear Regression and evaluate its performance.

CO2: Use different packages and frameworks to implement regression and classification algorithms.

Procedure

```
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.metrics import mean_squared_error from
sklearn.model_selection import train_test_split from
sklearn.linear_model import LinearRegression
california_housing=fetch_california_housing()
df=pd.DataFrame(data=california_housing.data,columns=california_housing.feature_names)
df['Target']=california_housing.target
x=df.drop('Target',axis=1)
y=df['Target']
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) predictions =
model.predict(x_test)
mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error: {mse}")
```

Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\pythonProject
Mean Squared Error: 0.555891598695244

Process finished with exit code 0
```

Result: : The program was executed successfully and the output was obtained. Thus, CO2 has been attained.

Experiment No.: 13

Aim: Program to implement Decision Tree Classification.

CO3: To what extend you are able to use different packages and frameworks to implement text classification using SVM and clustering using K-means?

Procedure

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split  from
sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.metrics import accuracy_score,classification_report from
matplotlib import pyplot as plt
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)
dt=DecisionTreeClassifier(max_depth=3)
dt.fit(x_train,y_train)
print(dt.predict(x_test))
D=dt.predict(x_test)
result=accuracy_score(y_test,D)
print("Accuracy= ",result)
cr=classification_report(y_test,D)
print("Classification Report: ",cr)
plt.figure(figsize=(15,20))
plot_tree(dt,filled=True,feature_names=iris.feature_names,class_names=iris.target_names)
plt.title("Decission Tree")
plt.show()
```
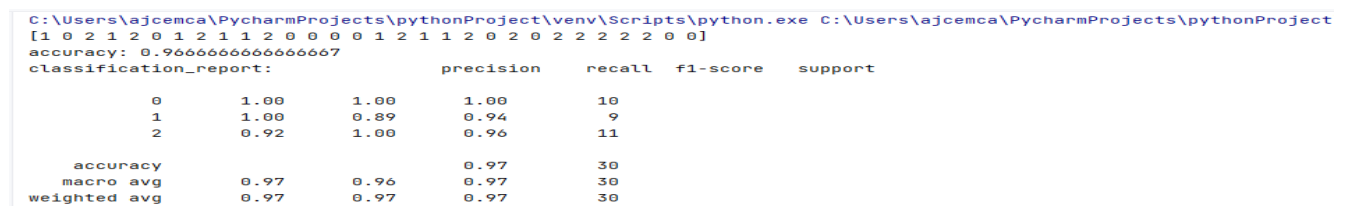
Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProject
[1 0 2 1 2 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
accuracy: 0.9666666666666667
classification_report:                    precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.89      0.94         9
           2       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30
```
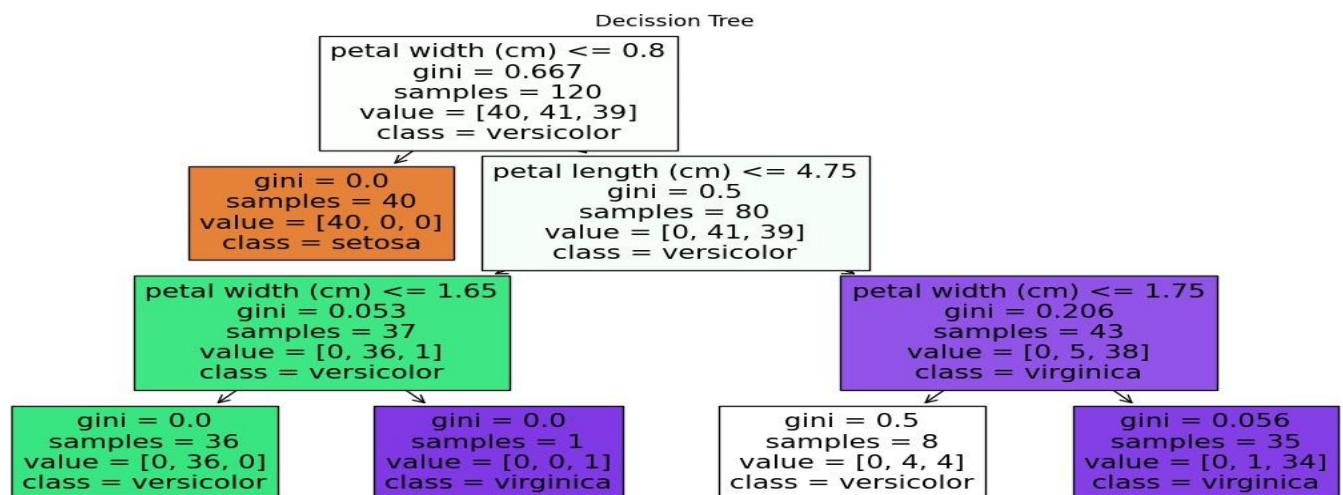
Decission Tree

Result: : The program was executed successfully and the output was obtained. Thus, CO3 has been attained.

Experiment No.: 14

Aim: Program to implement K-Means Classification.

CO3: To what extend you are able to use different packages and frameworks to implement text classification using SVM and clustering using K-means?

Procedure

```
from sklearn.datasets import load_iris from
sklearn.cluster import KMeans import
matplotlib.pyplot as plt
iris = load_iris() x =
iris.data
y = iris.target
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(x)
cluster_labels = kmeans.labels_
print(cluster_labels)
centroids = kmeans.cluster_centers_
print(centroids)
plt.scatter(x[:, 0], x[:, 1], c=cluster_labels, cmap='viridis', marker='o', edgecolors='black')
plt.scatter(centroids[:, 0], centroids[:, 1], marker="*", s=200, c='red', label='Centroids')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title('KMeans Cluster of Iris Dataset')
plt.legend()
plt.show()
```
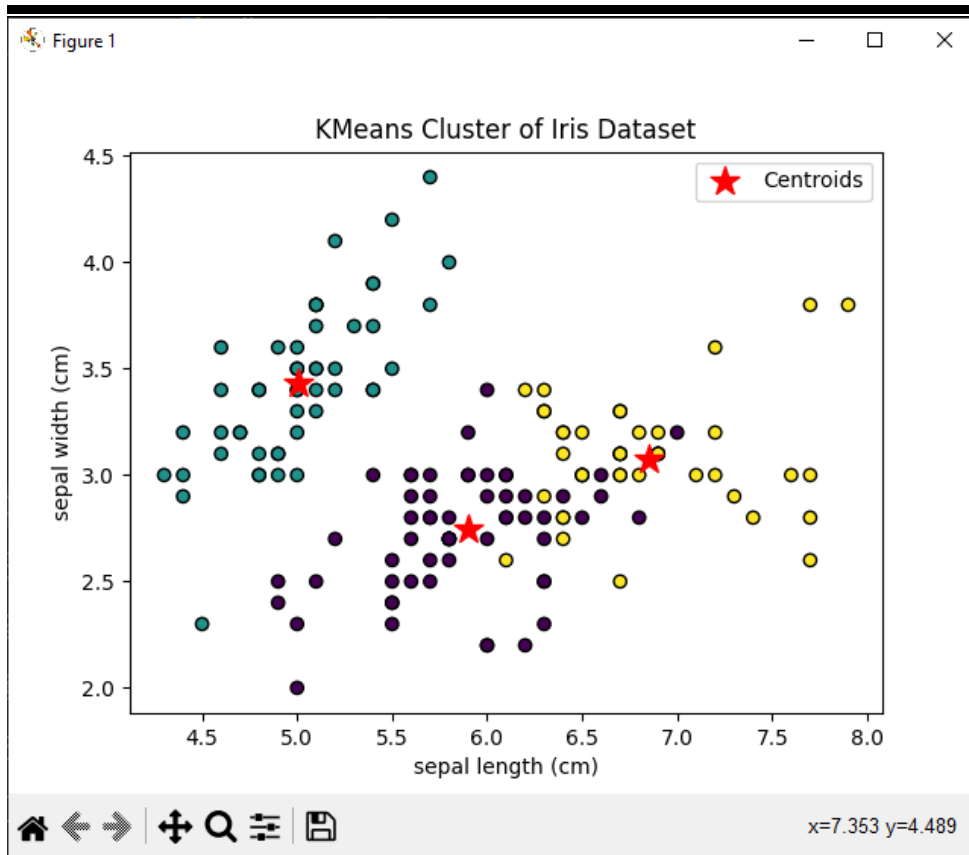
Output Screenshot

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
[[5.9016129  2.7483871  4.39354839 1.43387097]
 [5.006      3.428      1.462      0.246     ]
 [6.85       3.07368421 5.74210526 2.07105263]]
```

Result: : The program was executed successfully and the output was obtained. Thus, CO3 has been attained.
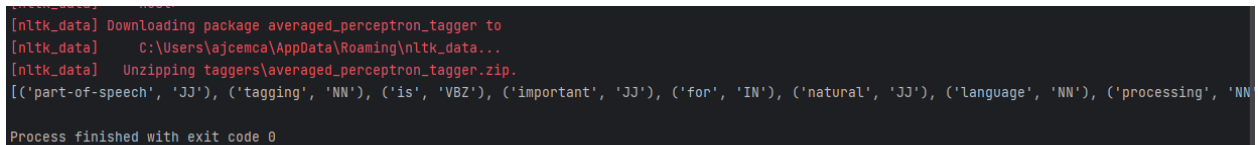
Experiment No.: 15

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

CO5: To what extend you are able to implement programs for web data mining and natural language processing using NLTK?

Procedure

```
import nltk
ffrom nltk import pos_tag
from nltk.tokenize import word_tokenize
'nltk.download(punkt)'
'nltk.download(averaged_perceptron_tagger)'
sentence="part-of-speech tagging is important for natural language processing"
words=word_tokenize(sentence)
pos_tag=pos_tag(words)
print(pos_tag)
```

Output Screenshot

```
[nltk_data]      C:\U...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]      C:\Users\ajcemca\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
[('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 was executed successfully and the output was obtained. Thus, CO5 has been attained.
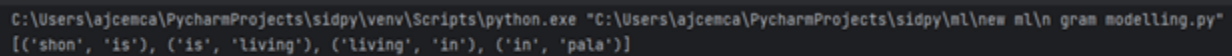
Experiment No.: 16

Aim: Program to implement N-Gram Modeling.

CO5: To what extend you are able to implement programs for web data mining and natural language processing using NLTK?

Procedure

```
from nltk import bigrams, word_tokenize
# Sample sentence
sentence = "shon is living in pala"
# Tokenize the sentence
words = word_tokenize(sentence)
# Generate bigrams
bigrams_list = list(bigrams(words))
# Display the result
print(bigrams_list)
```

Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\sidpy\venv\Scripts\python.exe "C:\Users\ajcemca\PycharmProjects\sidpy\ml\new ml\n gram modelling.py"
[('shon', 'is'), ('is', 'living'), ('living', 'in'), ('in', 'pala')]
```

Result: : The program was executed successfully and the output was obtained. Thus, CO5 has been attained.

Experiment No.: 17

Aim: Program to implement a simple web crawler.

CO5: To what extend you are able to implement programs for web data mining and natural language processing using NLTK?

Procedure

```
import requests
from bs4 import BeautifulSoup


def simple_web_crawler(url, max_depth=2):
 visited_urls = set()

 def crawl(url, depth)
if depth > max_depth or url in visited_urls:
                return

 print(f"Crawling: {url}")

        try:
                response = requests.get(url)
                visited_urls.add(url)


                if response.status_code == 200:
                   soup = BeautifulSoup(response.text, 'html.parser')
                   title = soup.title.string.strip() if soup.title else 'No title found'
                   print(f"Page Title: {title}")


                   for link in soup.find_all('a', href=True):
                      next_url = link['href']
                      crawl(next_url, depth + 1)


           except Exception as e:
              print(f"Error crawling {url}: {e}")


        crawl(url, depth=1)


    if __name__ == "__main__":
        start_url = "https://google.com"
        simple_web_crawler(start_url)
```
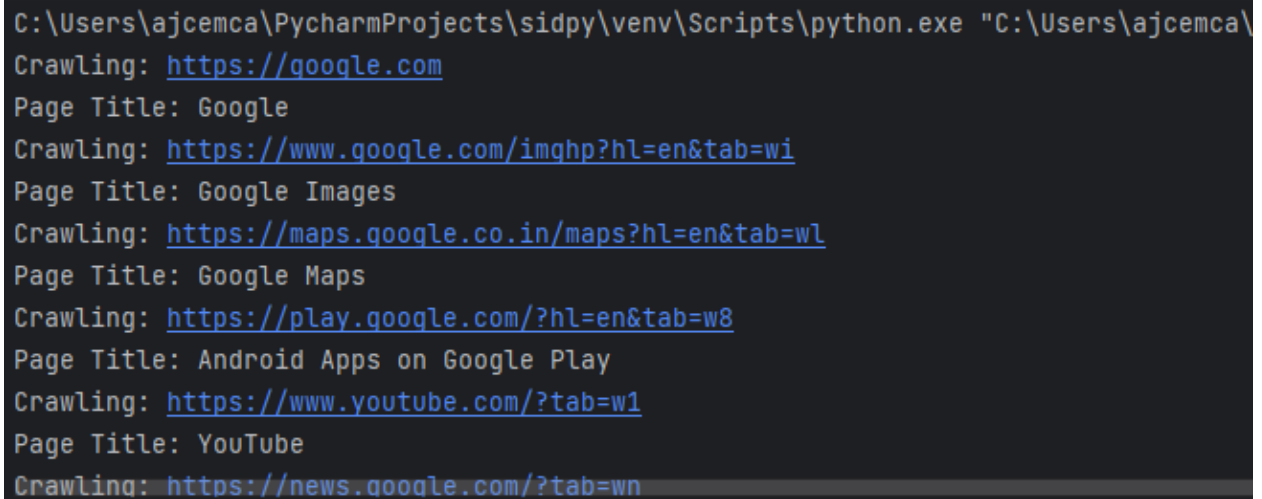
Output Screenshot

```
C:\Users\ajcemca\PycharmProjects\sidpy\venv\Scripts\python.exe "C:\Users\ajcemca\
Crawling: https://google.com
Page Title: Google
Crawling: https://www.google.com/imghp?hl=en&tab=wi
Page Title: Google Images
Crawling: https://maps.google.co.in/maps?hl=en&tab=wl
Page Title: Google Maps
Crawling: https://play.google.com/?hl=en&tab=w8
Page Title: Android Apps on Google Play
Crawling: https://www.youtube.com/?tab=w1
Page Title: YouTube
Crawling: https://news.google.com/?tab=wn
```

Result: : The program was executed successfully and the output was obtained. Thus, CO5 has been attained.

Experiment No.: 18

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

CO4: To what extend you are able to implement convolutional neural network algorithm using Keras framework?

Procedure

```
import numpy as np
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


        # Load the Iris dataset
        iris = load_iris()
        X = iris.data
        y = iris.target


        # Split the dataset 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)


        # Standardize the features
        scaler = StandardScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)


        # One-hot encode the target variable
        label_encoder = LabelEncoder()
        y_train = label_encoder.fit_transform(y_train)
        y_test = label_encoder.transform(y_test)


        # Create a simple feedforward neural network model
        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'))  # Output layer with 3 classes


        # Compile the model
        model.compile(optimizer='adam',
                loss='sparse_categorical_crossentropy',  # Use 'categorical_crossentropy' if y is one-
```

hot encoded
        metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=8, validation_split=0.1)

Evaluate the model on the test set
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f'Test accuracy: {test_acc}')

Output Screenshot

```
Epoch 45/50
14/14 [==============================] - 0s 5ms/step - loss: 0.0672 - accuracy: 0.9722 - val_loss: 0.4183 - val_accuracy: 0.9167
Epoch 46/50
14/14 [==============================] - 0s 4ms/step - loss: 0.0639 - accuracy: 0.9722 - val_loss: 0.4037 - val_accuracy: 0.9167
Epoch 47/50
14/14 [==============================] - 0s 4ms/step - loss: 0.0620 - accuracy: 0.9815 - val_loss: 0.4169 - val_accuracy: 0.9167
Epoch 48/50
14/14 [==============================] - 0s 4ms/step - loss: 0.0620 - accuracy: 0.9722 - val_loss: 0.4176 - val_accuracy: 0.9167
Epoch 49/50
14/14 [==============================] - 0s 4ms/step - loss: 0.0598 - accuracy: 0.9815 - val_loss: 0.4133 - val_accuracy: 0.9167
Epoch 50/50
14/14 [==============================] - 0s 4ms/step - loss: 0.0620 - accuracy: 0.9722 - val_loss: 0.4101 - val_accuracy: 0.9167
1/1 [==============================] - 0s 30ms/step - loss: 0.0369 - accuracy: 1.0000
Test accuracy: 1.0
```


Result: The program was executed successfully and the output was obtained. Thus, CO4 has been attained.