

# **20MCA241 – Data Science Lab**

*Lab Report Submitted By*

**GAYATHRI UNNIKRISHNAN**

**AJC22MCA-2045**

*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]

**2023-2024**

**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 **GAYATHRI UNNIKRISHNAN (AJC22MCA-2045)** 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. Navyamol K T  
**Lab In- Charge**

Rev. Fr. Dr. Rubin Thottupurathu Jose  
**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 extent 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 extent you are able to use different packages and frameworks to implement regression and classification algorithms?	Excellent/Very Good/Good/Satisfactory/Poor
CO3	To what extent 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 extent you are able to implement convolutional neural network algorithm using Keras framework?	Excellent/Very Good/Good/Satisfactory/Poor
CO5	To what extent you are able to implement programs for web data mining and natural language processing using NLTK?	Excellent/Very Good/Good/Satisfactory/Poor

# CONTENT

SL. NO.	LIST OF LAB EXPERIMENTS/EXERCISES	DATE	CO	PAGE NO
1	Program to perform matrix operations. Use numpy as the python library and perform the operation using built in functions.	25-09-23	CO1	1
2	Program to perform single value decomposition using numpy.	29-09-23	CO1	3
3	Program to perform data visualisation using python library matplotlib.	29-09-23	CO1	4
4	Program to implement KNN classification using any standard dataset available in the public domain and find the accuracy of algorithm (Iris Dataset)	10-10-23	CO2	5
5	Program to implement KNN classification using any standard dataset available in the public domain and find the accuracy of algorithm (Load Digits)	10-10-23	CO2	6
6	Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of algorithm (Iris Dataset)	31-10-23	CO2	7
7	Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of algorithm (Breast Cancer Dataset)	31-10-23	CO2	8
8	Give one dimensional dataset represented with numpy array. Write a program to calculate slope and intercept	10-11-23	CO2	9
9	Program to implement simple linear regression using any standard dataset available in the public domain and find r2 score.	07-11-23	CO2	10
10	Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance	10-11-23	CO2	12
11	Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm (Iris Dataset)	03-11-23	CO3	13
12	Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm (Breast Cancer Dataset)	03-11-23	CO3	15
13	Program to implement k-means clustering technique using any standard dataset available in the public domain (Iris Dataset)	21-11-23	CO3	17
14	Program to implement k-means clustering technique using any standard dataset available in the public domain (Breast Cancer Dataset)	21-11-23	CO3	19
15	Program to implement text classification using support vector machine.	30-11-23	CO3	22
16	Program on artificial neural network to classify images from any standard dataset in the public domain using Keras framework.	01-12-23	CO4	24

17	Program to implement a simple web crawler using requests library	06-12-23	CO5	26
18	Program to implement a simple web crawler and parse the content using BeautifulSoup.	06-12-23	CO5	27
19	Implement problems on natural language processing - Part of Speech tagging, N-gram & smoothening and Chunking using NLTK	07-12-23	CO5	28

---

**Experiment No.: 1**

**Aim:** Write a program to perform matrix operations. Use Numpy as the python library and perform the operations using built in functions in Numpy.

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

**Procedure:**

```
import numpy as np

def input_matrix(matrix_name):
    rows = int(input(f"Enter the no of rows of {matrix_name}: "))
    cols = int(input(f"Enter the no of columns of {matrix_name}: "))
    matrix=[]
    print("Enter the elements:")
    for i in range(rows):
        rows=[]
        for j in range(cols):
            element=int(input(f"Enter the element at row {i+1},column {j+1}:"))
            rows.append(element)
        matrix.append(rows)
    return np.array(matrix)

matrix1 = input_matrix("matrix1")
matrix2 = input_matrix("matrix2")
sum = np.add(matrix1, matrix2)
print("Addition=", sum)
diff = np.subtract(matrix1, matrix2)
print("Subtraction=", diff)
prod = np.multiply(matrix1, matrix2)
print("Multiplication=", prod)
div = np.divide(matrix1, matrix2)
print("Division=", div)
transp = np.transpose(matrix1)
print("Transpose=", transp)
dot = np.dot(matrix1, matrix2)
print("Dot product=", dot)
```

---

## **Output Screenshot**

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProject\gayathri\exp1.py
Enter the no of rows of matrix1: 2
Enter the no of columns of matrix1: 2
Enter the elements:
Enter the element at row 1,column 1:1
Enter the element at row 1,column 2:2
Enter the element at row 2,column 1:3
Enter the element at row 2,column 2:4
Enter the no of rows of matrix2: 2
Enter the no of columns of matrix2: 2
Enter the elements:
Enter the element at row 1,column 1:5
Enter the element at row 1,column 2:6
Enter the element at row 2,column 1:7
Enter the element at row 2,column 2:8
Addition:
[[ 6  8]
 [10 12]]
Subtraction:
[[-4 -4]
 [-4 -4]]
Multiplication:
[[ 5 12]
 [21 32]]

Division:
[[0.2      0.33333333]
 [0.42857143 0.5      ]]
Transpose:
[[1 3]
 [2 4]]
Dot product:
[[19 22]
 [43 50]]
```

## **Result:**

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

---

## **Experiment No.: 2**

**Aim:** Program to perform single value decomposition(SVD) using python Numpy.

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

### **Procedure:**

```
import numpy as np
matrix = np.array([[5, 6, 4],
                  [2, 5, 6],
                  [3, 5, 6]])

U, S, VT = np.linalg.svd(matrix)
print("Matrix U: ")
print(U)

print("Matrix S: ")
print(np.diag(S))
print("Matrix VT: ")
print(VT)

#reconstruct old matrix
recnstrct = np.dot(U,np.dot(np.diag(S),VT))
print("Old Matrix: ")
print(recnstrct)
```

### **Output Screenshot**



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProject\gayathri\exp2.py
U Matrix:
[[-0.59482308  0.7878662 -0.15953794]
 [-0.55395727 -0.54556995 -0.6288758 ]
 [-0.58250909 -0.28569264  0.76096181]]
S Matrix:
[[14.28896808  0.          0.         ]
 [ 0.          2.76798539  0.         ]
 [ 0.          0.          0.40453427]]
VT Matrix:
[[-0.40797608 -0.64744146 -0.64371972]
 [ 0.71933659  0.2062454 -0.6633383 ]
 [ 0.56223695 -0.73367731  0.38158514]]
Reconstructed Matrix:
[[5.  6.  4.]
 [2.  5.  6.]
 [3.  5.  6.]]

Process finished with exit code 0
```

### **Result:**

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



## **Experiment No.: 3**

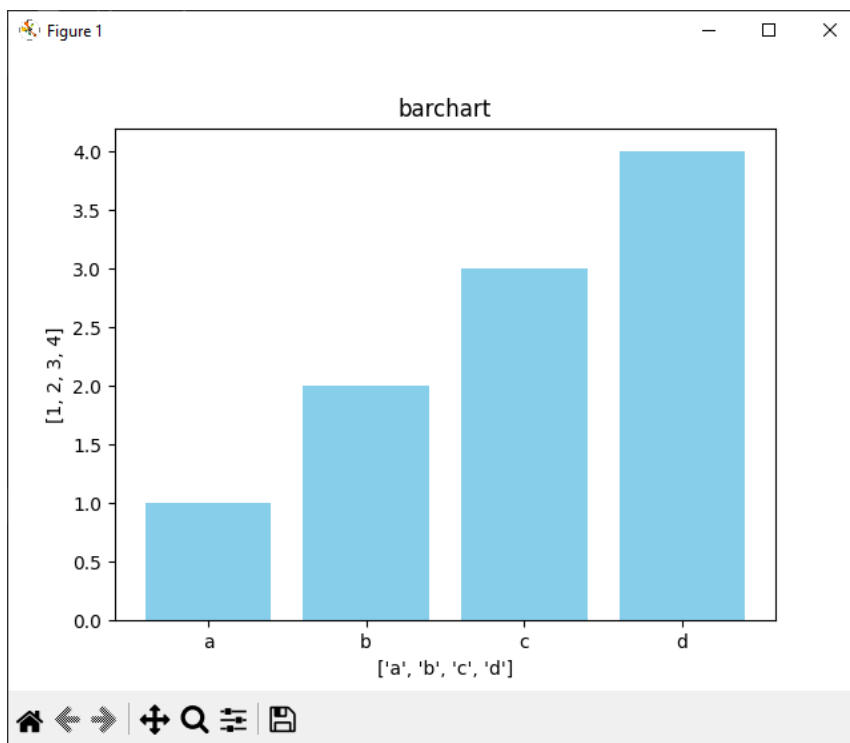
**Aim:** Program to perform data visualisation using the python library matplotlib.

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

### **Procedure:**

```
import matplotlib.pyplot as plt
categories=["a","b","c","d"]
values=[1,2,3,4]
plt.bar(categories,values,color='skyblue')
plt.xlabel(categories)
plt.ylabel(values)
plt.title("barchart")
plt.show()
```

### **Output Screenshot**



### **Result:**

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

---

## **Experiment No.: 4**

**Aim:** Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm. (Iris Dataset)

**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.: 5**

**Aim:** Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm. (Load Digits)

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

### **Procedure:**

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
digits=load_digits()
x=digits.data
y=digits.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)))
P=knn.predict(x_test)
R=accuracy_score(y_test,P)
print("Accuracy= ",R)
```

### **Output Screenshot**

```
C:\Users\ajcemca\PycharmProjects\DSML\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\DSML\KNNsample.py
[6 9 3 7 2 1 5 2 5 2 1 9 4 0 4 2 3 7 8 8 4 3 9 7 5 6 3 5 6 3 4 9 1 4 4 6 9
 4 7 6 6 9 1 3 6 1 3 0 6 5 5 1 9 5 6 0 9 0 0 1 0 4 5 2 4 5 7 0 7 5 9 9 5 4
 7 0 4 5 5 9 9 0 2 3 8 0 6 4 4 9 1 2 8 3 5 2 9 0 4 4 4 3 5 3 1 3 5 9 4 2 7
 7 4 4 1 9 2 7 8 7 2 6 9 4 0 7 2 7 5 8 7 5 7 7 0 6 6 4 2 8 0 9 4 6 9 9 6 9
 0 3 5 6 6 0 6 4 3 9 3 4 7 2 9 0 4 5 3 6 5 9 9 8 4 2 1 3 7 7 2 2 3 9 8 0 3
 2 2 5 6 9 9 4 1 5 4 2 3 6 4 8 5 9 5 7 1 9 4 8 1 5 4 4 9 6 1 8 6 0 4 5 2 7
 4 6 4 5 6 0 3 2 3 6 7 1 5 1 4 7 6 8 8 5 5 1 6 2 8 8 9 5 7 6 2 2 2 3 4 8 8
 3 6 0 9 7 7 0 1 0 4 5 1 5 3 6 0 4 1 0 0 3 6 5 9 7 3 5 5 9 9 8 5 3 3 2 0 5
 8 3 4 0 2 4 6 4 3 4 5 0 5 2 1 3 1 4 1 1 7 0 1 5 2 1 2 8 7 0 6 4 8 8 5 1 8
 4 5 8 7 9 8 6 0 6 2 0 7 9 1 9 5 2 7 7 1 8 7 4 3 8 3 5]
accuracy: 0.9833333333333333

Process finished with exit code 0
|
```

### **Result:**

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

## **Experiment No.: 6**

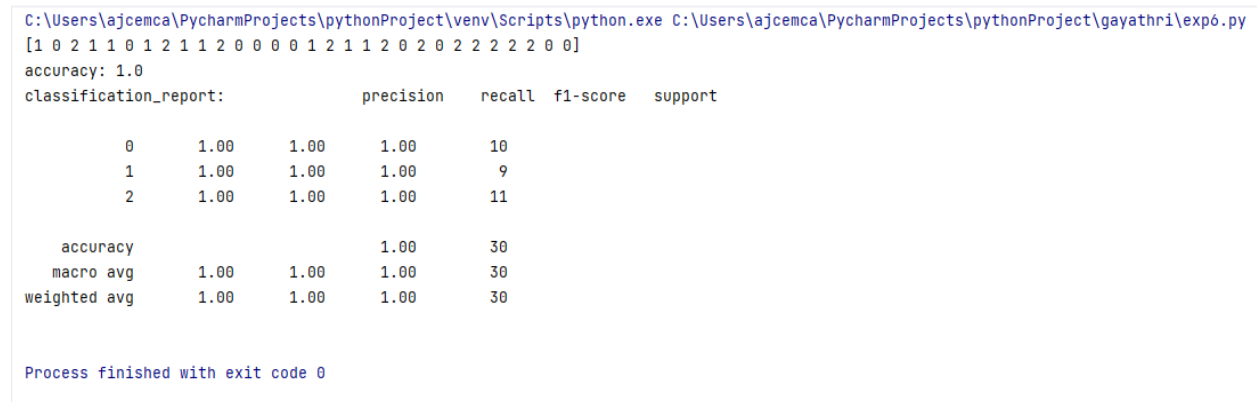
**Aim:** Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

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

### **Procedure:**

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
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)
clf=GaussianNB()
clf.fit(x_train,y_train)
print(clf.predict(x_test))
V=clf.predict(x_test)
result=accuracy_score(y_test,V)
print("Accuracy= ",result)
```

### **Output Screenshot**



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProject\gayathri\exp6.py
[1 0 2 1 1 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: 1.0
classification_report:
              precision    recall  f1-score   support

     0           1.00      1.00      1.00         10
     1           1.00      1.00      1.00          9
     2           1.00      1.00      1.00         11

   accuracy          1.00      1.00      1.00         30
  macro avg          1.00      1.00      1.00         30
 weighted avg          1.00      1.00      1.00         30

Process finished with exit code 0
```

### **Result:**

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

## **Experiment No.: 7**

**Aim:** Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

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

### **Procedure:**

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
bc=load_breast_cancer()
x=bc.data
y=bc.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
gnb=GaussianNB()
gnb.fit(x_train,y_train)
print(gnb.predict(x_test))
G=gnb.predict(x_test)
result=accuracy_score(y_test,G)
print("Accuracy= ",result)
cr=classification_report(y_test,G)
print("/n Classification Report: ",cr)
```

### **Output Screenshot**

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProject\gayathri\exp7.py
[1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 0
 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0
 1 1 0]
accuracy: 0.9736842105263158
classification_report:              precision    recall  f1-score   support

      0       1.00      0.93      0.96         43
      1       0.96      1.00      0.98         71

   accuracy                   0.97         114
  macro avg                   0.98         114
weighted avg                   0.97         114

Process finished with exit code 0
```

### **Result:**

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

## **Experiment No.: 8**

**Aim:** Given a one dimensional data represented with Numpy array. Write a program to calculate slope and intercept.

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

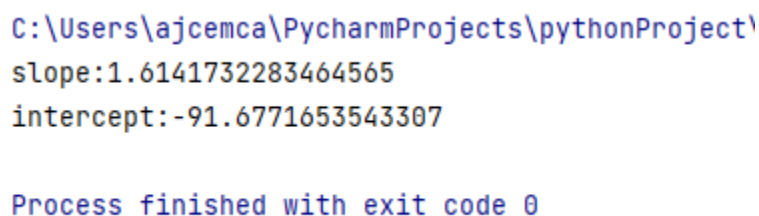
### **Procedure:**

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

x_value = np.array([64,75,68,73,78,82,76,85,71,88]).reshape(-1,1)
y_value = np.array([17,27,15,24,39,44,30,48,19,47])

model=LinearRegression()
model.fit(x_value,y_value)
slope=model.coef_[0]
intercept=model.intercept_
print(f"Slope: {slope}")
print(f"Intercept: {intercept}")
```

### **Output Screenshot**



```
C:\Users\ajcemca\PycharmProjects\pythonProject\
slope:1.6141732283464565
intercept:-91.6771653543307

Process finished with exit code 0
```

### **Result:**

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

---

## **Experiment No.: 9**

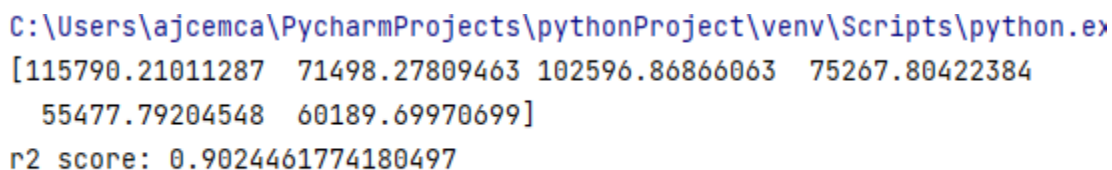
**Aim:** Program to implement Simple Linear Regression using any standard dataset available in public domain and find the R2 score.

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

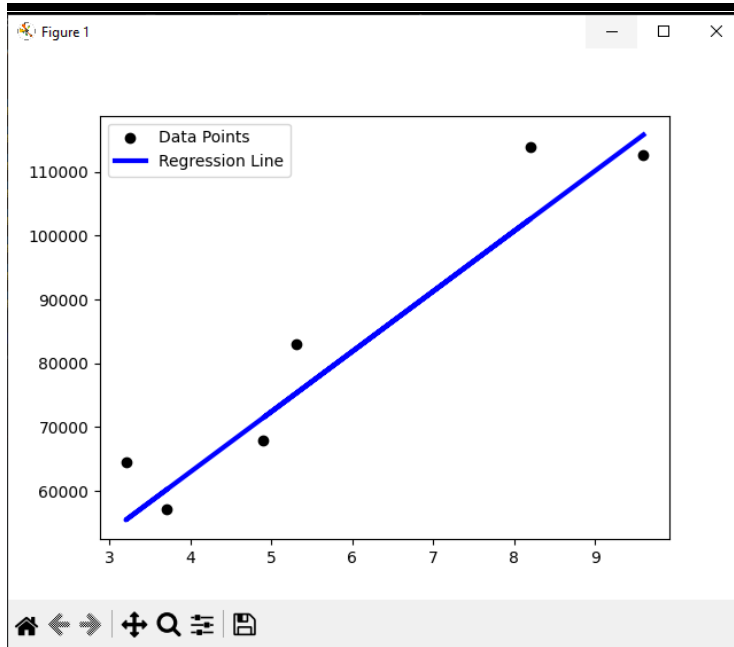
### **Procedure:**

```
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.exe
[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<sub>2</sub> has been attained.



**Experiment No.: 10**

**Aim:** Program to implement Multiple Linear Regression using any standard dataset available in public domain.

**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.: 11**

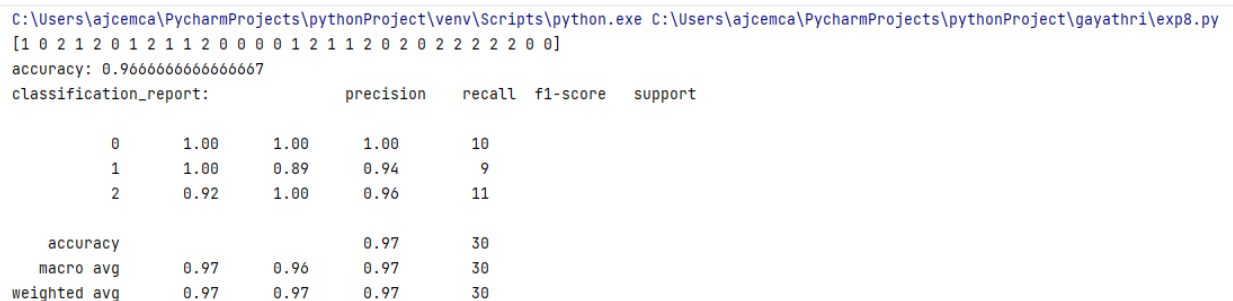
**Aim:** Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.

**CO3:** 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()
```

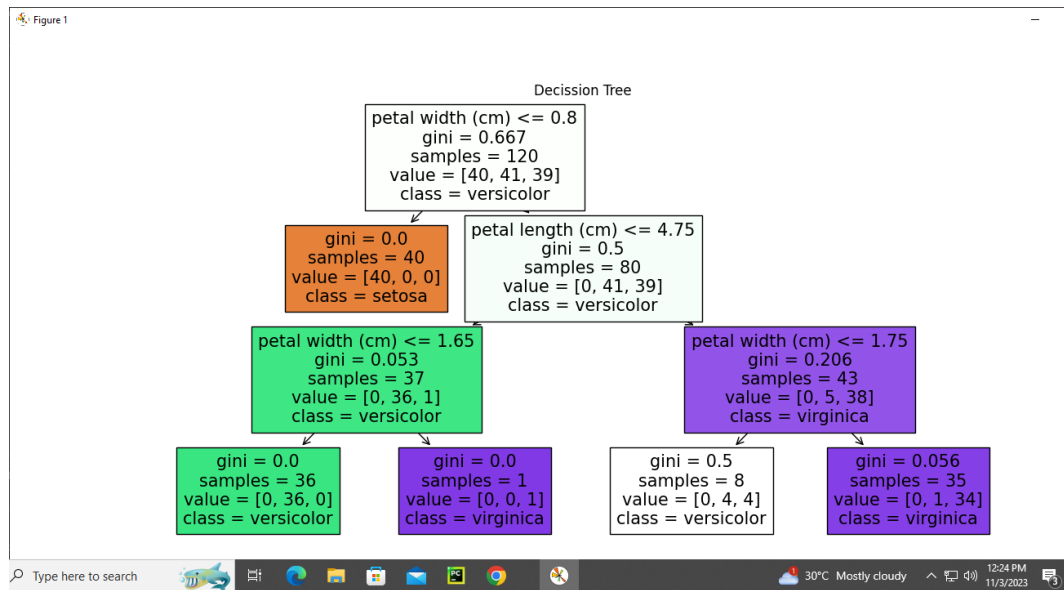
### **Output Screenshot**



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProject\gayathri\exp8.py
[1 0 2 1 2 0 1 2 1 1 2 0 0 0 1 2 1 1 2 0 2 0 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
  macro avg          0.97      0.96      0.97        30
 weighted avg          0.97      0.97      0.97        30
```



### **Result:**

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

## **Experiment No.: 12**

**Aim:** Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.

**CO3:** Use different packages and frameworks to implement text classification using SVM and clustering using k-means

### **Procedure:**

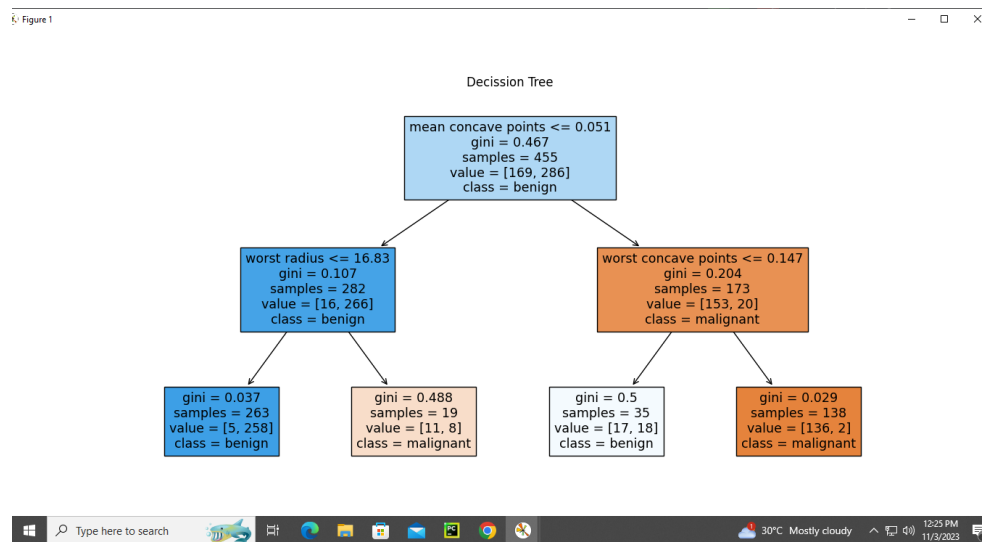
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report
from matplotlib import pyplot as plt
bc=load_breast_cancer()
x=bc.data
y=bc.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
dt=DecisionTreeClassifier(max_depth=2)
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=(15,10))
plot_tree(dt,filled=True,feature_names=bc.feature_names,class_names=bc.target_names)
plt.title("Decission Tree")
plt.show()
```

### **Output Screenshot**

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProject\gayathri\exp9.py
[1 0 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0
 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 0 1 0
 1 1 0]
accuracy: 0.9298245614035088
classification_report:              precision    recall  f1-score   support

      0       0.95      0.86      0.90         43
      1       0.92      0.97      0.95         71

   accuracy              0.93         114
  macro avg              0.93      0.92      0.92         114
 weighted avg              0.93      0.93      0.93         114
```



### Result:

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

**Experiment No.: 13**

**Aim:** Program to implement k-means clustering technique using any standard dataset available in the public domain

**CO3:** 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

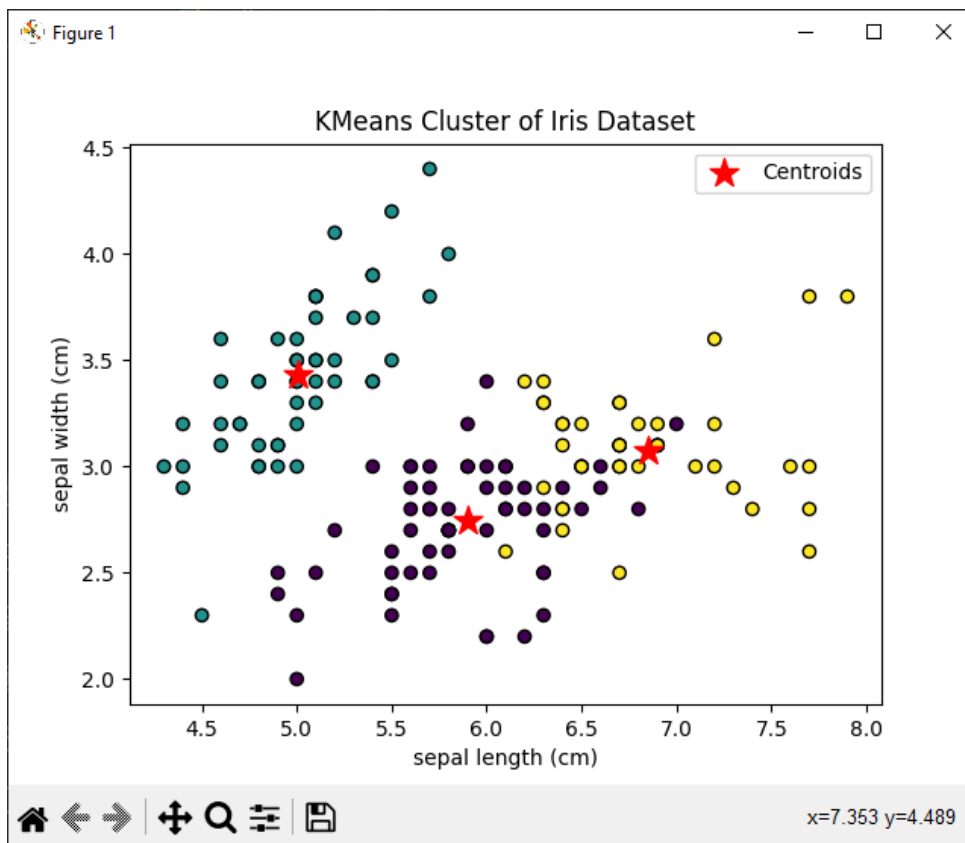
iris = load_iris()
x = iris.data
y = iris.target

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



**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 clustering technique using any standard dataset available in the public domain

**CO3:** Use different packages and frameworks to implement text classification using SVM and clustering using k-means

**Procedure:**

```
from sklearn.datasets import load_breast_cancer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

bc = load_breast_cancer()
x = bc.data
y = bc.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)

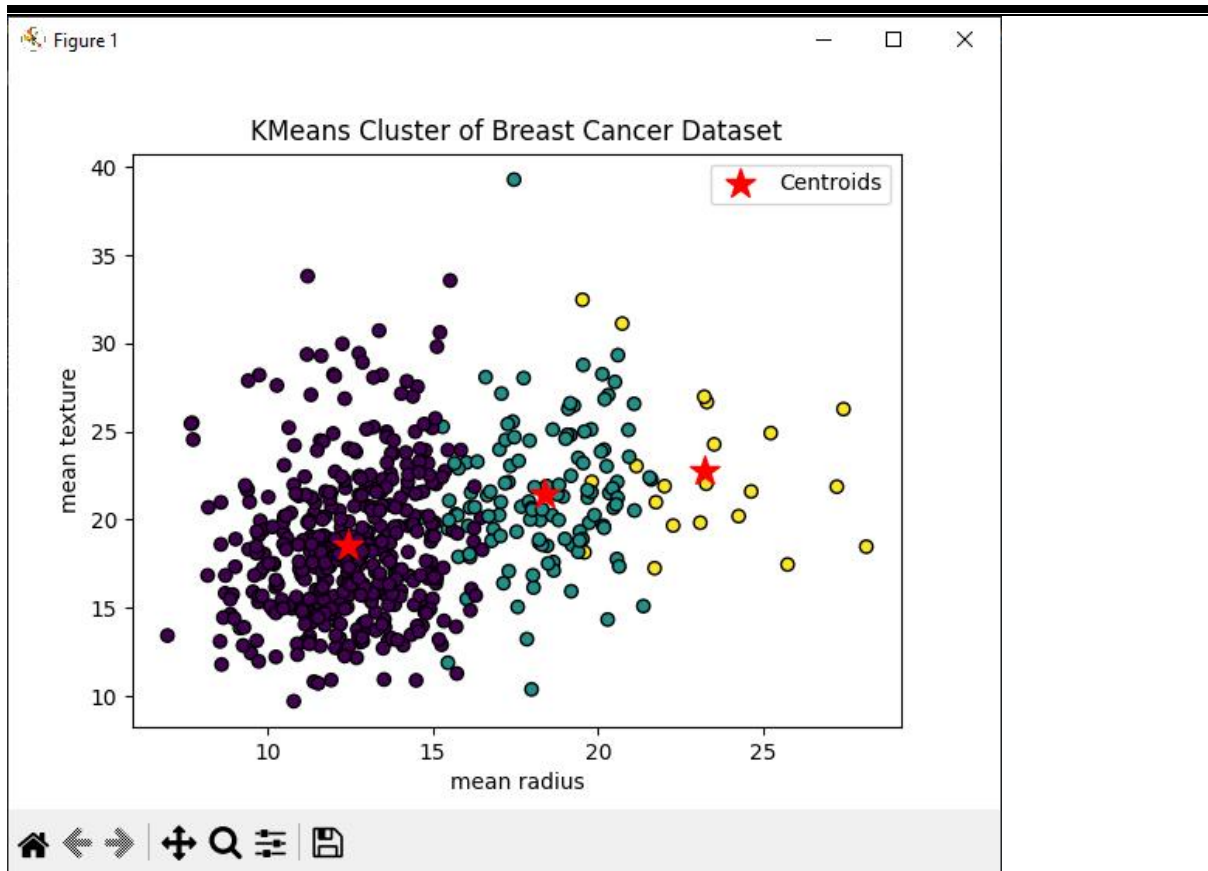
plt.scatter(x[:, 0], x[:, 1], c=cluster_labels, cmap='viridis', marker='o', edgecolors='black')
plt.scatter(centroids[:, 0], centroids[:, 1], marker="*", s=200, c='red', label='Centroids')
plt.xlabel(bc.feature_names[0])
plt.ylabel(bc.feature_names[1])
plt.title('KMeans Cluster of Breast Cancer Dataset')
plt.legend()
plt.show()
```



## Output Screenshot

```
[1 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 1 1 2 0 0 0 0 2 1 1 0 1 1 1 1 0 1 1 1 0
0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
0 1 0 1 1 0 0 0 2 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 2 0 0
0 0 0 0 0 0 0 1 1 0 1 2 0 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 1 2 0 2 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 2 1 1 0 0
0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 2 0 0 0 0 1 0 0 1 0 2 1 0 0 0 0 1 2 0 0
0 1 0 0 0 0 0 0 1 0 0 1 0 0 2 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 0 1 0 1
0 1 1 1 0 1 2 0 0 0 0 0 0 2 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0
0 0 1 0 1 0 2 0 0 0 1 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 2 2
1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0
1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 2 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 1 1 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 2 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 0]
```

```
[[1.24468918e+01 1.85046588e+01 8.03803294e+01 4.86458118e+02
9.48157176e-02 9.04881882e-02 6.08800016e-02 3.25494682e-02
1.77951765e-01 6.35771765e-02 3.00681647e-01 1.21837294e+00
2.12940400e+00 2.32080188e+01 7.17541647e-03 2.33490235e-02
2.84143873e-02 1.05452329e-02 2.07137600e-02 3.75171835e-03
1.38918094e+01 2.45948235e+01 9.09125412e+01 6.04658353e+02
1.29845529e-01 2.21074000e-01 2.14822228e-01 8.97035082e-02
2.82468471e-01 8.32831059e-02]
[1.83820325e+01 2.14148780e+01 1.21238537e+02 1.05796098e+03
1.00221870e-01 1.40414797e-01 1.58604959e-01 9.06387805e-02
1.91033333e-01 6.06883740e-02 6.40926016e-01 1.20443577e+00
4.50100813e+00 7.53708943e+01 6.57197561e-03 3.09567967e-02
4.08994309e-02 1.53647805e-02 2.00486992e-02 3.93508943e-03
2.22162602e+01 2.86411382e+01 1.47833333e+02 1.52278862e+03
1.39408780e-01 3.45692358e-01 4.26761789e-01 1.81023984e-01
3.15549593e-01 8.64585366e-02]
[2.32147619e+01 2.27285714e+01 1.55066667e+02 1.70276190e+03
1.05001429e-01 1.73405714e-01 2.44971429e-01 1.35852381e-01
1.88309524e-01 5.93747619e-02 1.13901429e+00 1.25883333e+00
8.19842857e+00 1.81798571e+02 7.06723810e-03 3.64780952e-02
4.95609524e-02 1.62100000e-02 1.99633333e-02 3.84780952e-03
2.95500000e+01 3.02228571e+01 2.00490476e+02 2.70328571e+03
1.42195238e-01 3.90485714e-01 5.27814286e-01 2.29571429e-01
2.94823810e-01 8.26404762e-02]]
```

**Result:**

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

---

## **Experiment No.: 15**

**Aim:** Program to implement text classification using support vector machine.

**CO3:** Use different packages and frameworks to implement text classification using SVM and clustering using k-means

### **Procedure:**

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score ,classification_report

categories=['alt.atheism','soc.religion.christian','comp.graphics','sci.med']
twenty_train=fetch_20newsgroups(subset='train',categories=categories,shuffle=
True,random_state=42)

vectorizer=TfidfVectorizer()
X_train_tfidf=vectorizer.fit_transform(twenty_train.data)
y_train = twenty_train.target

x_train,x_test,y_train,y_test=train_test_split(X_train_tfidf,y_train,test_siz
e=0.2,random_state=42)

svm_classifier=SVC(kernel='linear',random_state=42)
svm_classifier.fit(x_train,y_train)
predictions=svm_classifier.predict(x_test)
accuracy=accuracy_score(y_test,predictions)
classification=classification_report(y_test,predictions,target_names=twenty_t
rain.target_names)

print("Accuracy",accuracy)
print("classification_report:",classification)

new_data=["I have a question about computer graphics","This is a medical
related topic"]
x_new_tfidf=vectorizer.transform(new_data)
```

---

```
new_predictions=svm_classifier.predict(x_new_tfidf)
```

```
for i,text in enumerate(new_data):
    predicted_category=twenty_train.target_names[new_predictions[i]]
    print("Predicted Cate",predicted_category)
```

## **Output Screenshot**

```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProject\gayathri\test_classification.py
Accuracy 0.9601769911504425
classification_report:
              precision    recall  f1-score   support

   alt.atheism       0.96      0.95      0.96         86
  comp.graphics       0.91      1.00      0.96        107
     sci.med         0.98      0.95      0.97        132
 soc.religion.christian 0.98      0.94      0.96        127

   accuracy                   0.96        452
  macro avg       0.96      0.96      0.96        452
 weighted avg       0.96      0.96      0.96        452

Predicted Cate comp.graphics
Predicted Cate sci.med
```

## **Result:**

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

---

## **Experiment No.: 16**

**Aim:** Program on artificial neural network to classify images from any standard dataset in the public domain using Keras framework.

**CO4:** Implement convolutional neural network algorithm using Keras framework.

### **Procedure:**

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical
# Load the MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# Normalize pixel values to be between 0 and 1
X_train = X_train / 255.0
X_test = X_test / 255.0
# Flatten the images (convert 28x28 images to 1D vectors)
X_train = X_train.reshape(-1, 28 * 28)
print(X_train)
X_test = X_test.reshape(-1, 28 * 28)
print(X_test)
# One-hot encode the target labels
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
print(y_test)
# Create a simple feedforward neural network model
model=Sequential([
Dense(128, activation='relu', input_shape=(28 * 28,)),
Dense(68, activation='relu'),
Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(X_train,y_train, epochs=5 , batch_size=32, validation_split=0.2)
loss, accuracy= model.evaluate(X_test,y_test)
print(accuracy)
```

---

## **Output Screenshot**

```
1500/1500 [=====] - 3s 2ms/step - loss: 0.2654 - accuracy: 0.9220 - val_loss: 0.1610 - val_accuracy: 0.9513
Epoch 2/5
1500/1500 [=====] - 2s 2ms/step - loss: 0.1121 - accuracy: 0.9662 - val_loss: 0.1144 - val_accuracy: 0.9669
Epoch 3/5
1500/1500 [=====] - 2s 1ms/step - loss: 0.0782 - accuracy: 0.9761 - val_loss: 0.1126 - val_accuracy: 0.9658
Epoch 4/5
1500/1500 [=====] - 2s 1ms/step - loss: 0.0588 - accuracy: 0.9810 - val_loss: 0.1107 - val_accuracy: 0.9696
Epoch 5/5
1500/1500 [=====] - 2s 1ms/step - loss: 0.0441 - accuracy: 0.9861 - val_loss: 0.1032 - val_accuracy: 0.9725
313/313 [=====] - 0s 956us/step - loss: 0.0967 - accuracy: 0.9715
0.9714999794960022
```

## **Result:**

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

---

## **Experiment No.: 17**

**Aim:** Program to implement a simple web crawler using requests library.

**CO5:** Implement programs for web data mining and natural language processing using NLTK.

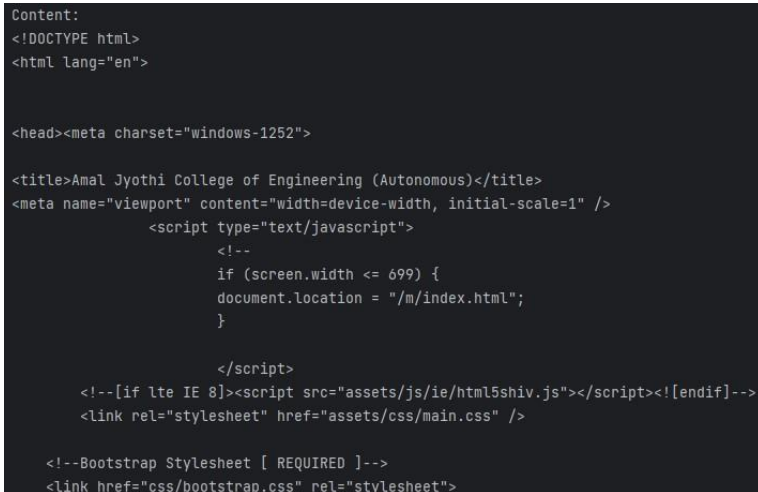
### **Procedure:**

import requests

```
def simple_scraper(url):  
    response=requests.get(url)  
    if response.status_code==200:  
        print("Content:")  
        print(response.text)  
    else:  
        print("Failed to fetch the page. Status code:", response.status_code)
```

```
url_to_scrap="https://ajce.in"  
simple_scraper(url_to_scrap)
```

### **Output Screenshot**



```
Content:  
<!DOCTYPE html>  
<html lang="en">  
  
<head><meta charset="windows-1252">  
  
<title>Amal Jyothi College of Engineering (Autonomous)</title>  
<meta name="viewport" content="width=device-width, initial-scale=1" />  
    <script type="text/javascript">  
        <!--  
            if (screen.width <= 699) {  
                document.location = "/m/index.html";  
            }  
        </script>  
    <!--[if lte IE 8]><script src="assets/js/ie/html5shiv.js"></script><![endif]-->  
    <link rel="stylesheet" href="assets/css/main.css" />  
  
    <!--Bootstrap Stylesheet [ REQUIRED ]-->  
    <link href="css/bootstrap.css" rel="stylesheet">
```

### **Result:**

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

---

## **Experiment No.: 18**

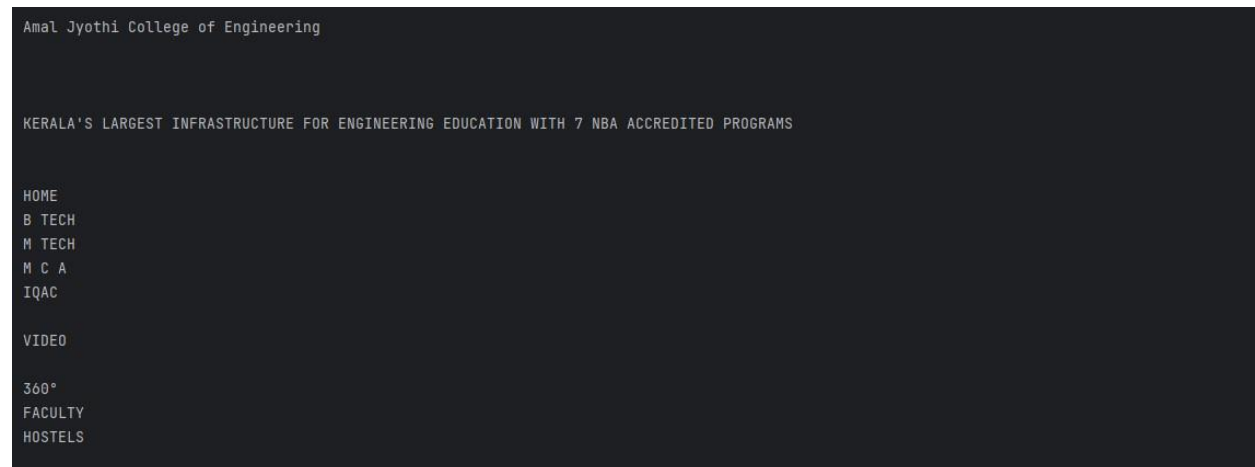
**Aim:** Program to implement a simple web crawler and parse the content using BeautifulSoup.

**CO5:** Implement programs for web data mining and natural language processing using NLTK.

### **Procedure:**

```
import requests
from bs4 import BeautifulSoup
def simple_scraper(url):
    response=requests.get(url)
    if response.status_code==200:
        soup=BeautifulSoup(response.content, 'html.parser')
        print("Title:",soup.title.string)
        print("Content:")
        print(soup.get_text())
    else:
        print("Failed to fetch the page. Status code:", response.status_code)
url_to_scrap="https://ajce.in"
simple_scraper(url_to_scrap)
```

### **Output Screenshot**



### **Result:**

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



**Experiment No.: 19**

**Aim:** Implement problems on natural language processing - Part of Speech tagging, N-gram & smoothing and Chunking using NLTK

**CO5:** Implement programs for web data mining and natural language processing using NLTK.

**Procedure:**

```
import nltk

nltk.download('brown')

nltk.download('punkt')

nltk.download('averaged_perceptron_tagger')

from nltk.tokenize import word_tokenize

from nltk.util import ngrams

from nltk.corpus import brown

from nltk.chunk import RegexpParser

sentence = "The quick brown fox jumps over the lazy dog"

tokens = word_tokenize(sentence)

print(tokens)

pos_tags = nltk.pos_tag(tokens)

print("Part-of-Speech Tagging: ")

print(pos_tags)

text = brown.words(categories='news')[:1000]

bigrams = list(ngrams(text, 2))

freq_dist = nltk.FreqDist(bigrams)

print("\n N-gram Analysis (Bigrams with Smoothing): ")

for bigram in bigrams:

    print(f"{bigram}: {freq_dist[bigram]}")

tagged_sentence = nltk.pos_tag(word_tokenize("The quick brown fox jumps over the lazy dog"))

grammar = r"NP: {<DT>?<JJ>*<NN>}"

cp = RegexpParser(grammar)
```

---

```
result = cp.parse(tagged_sentence)

print("\n Chunking with Regular Expressions and POS tags: ")

print(result)
```

## **Output Screenshot**

```
['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
Part-of-Speech Tagging:
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy', 'JJ'), ('dog', 'NN')]

N-gram Analysis (Bigrams with Smoothing):
('The', 'Fulton'): 1
('Fulton', 'County'): 6
('County', 'Grand'): 1
('Grand', 'Jury'): 1
('Jury', 'said'): 1
('said', 'Friday'): 1
('Friday', 'an'): 1
('an', 'investigation'): 1
('investigation', 'of'): 1
('of', 'Atlanta's'): 1
```

## **Result:**

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