Experiment No.: 1

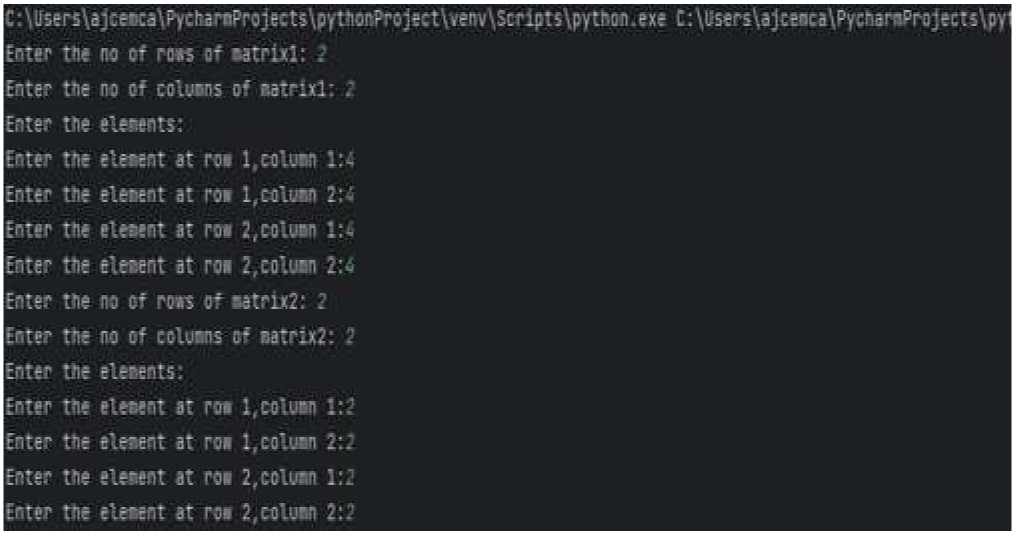
Aim: Program to perform matrix operations. Use NumPy as the python library and perform the operations using built in functions in NumPy.

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

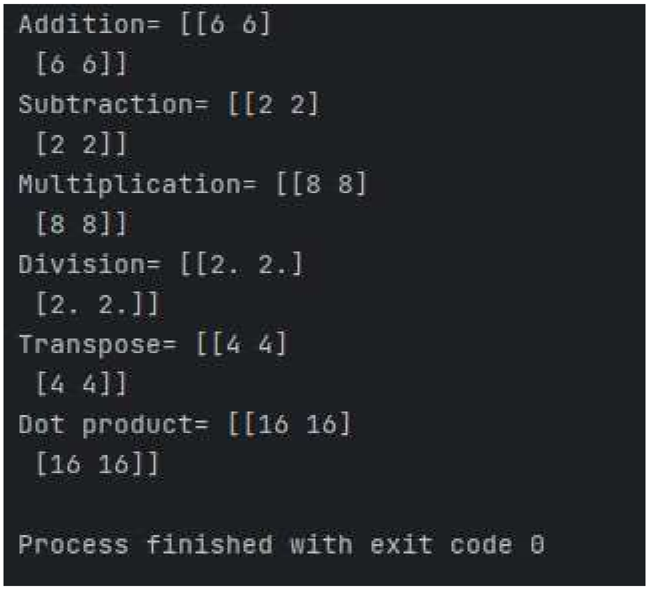
Procedure:

```
import numpy as np
def input_matrix(ourmatrix):
    r = int(input(f"Enter the no of rows for {ourmatrix}:"))
    c = int(input(f"Enter the no of columns for {ourmatrix}:"))
    matrix=[]
    print("Enter the elements:")
    for i in range(r):
        r=[]
        for j in range(c):
            elements=int(input(f"enter the element at row{i+1},colomn{j+1}"))
            r.append(elements)
            matrix.append(r)
    return np.array(matrix)
matrix1=input_matrix("matrix1")
input_matrix(matrix1)
matrix2=input_matrix("matrix2")
input_matrix(matrix2)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pyt
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 1:4
Enter the element at row 1,column 2:4
Enter the element at row 2,column 1:4
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 1:2
Enter the element at row 1,column 2:2
Enter the element at row 2,column 1:2
Enter the element at row 2,column 2:2
```



```
Addition= [[6 6]
 [6 6]]
Subtraction= [[2 2]
 [2 2]]
Multiplication= [[8 8]
 [8 8]]
Division= [[2. 2.]
 [2. 2.]]
Transpose= [[4 4]
 [4 4]]
Dot product= [[16 16]
 [16 16]]

Process finished with exit code 0
```

Result:

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

Experiment No.: 2

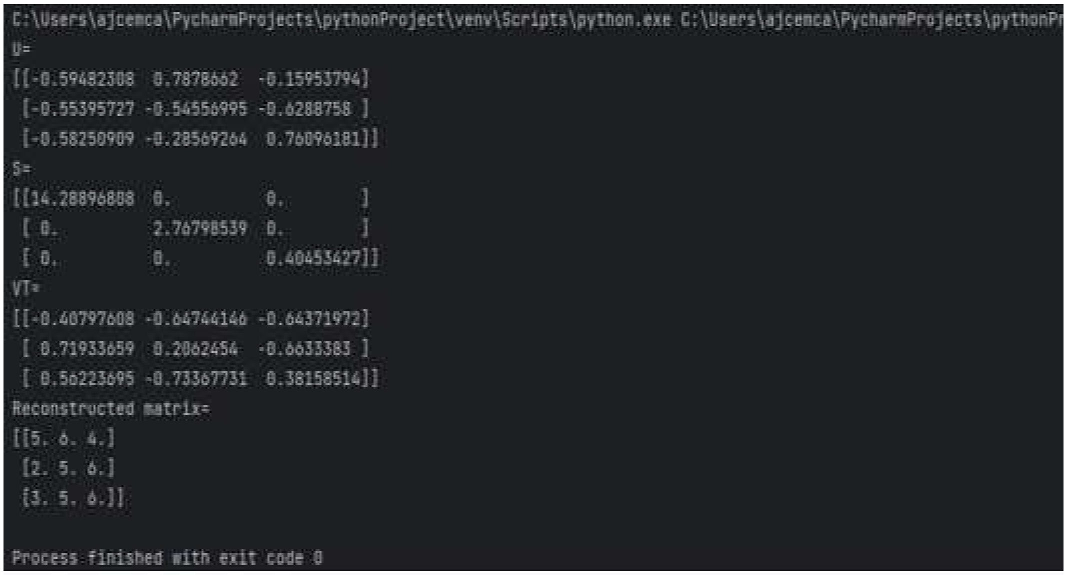
Aim: Program to perform single value decomposition using NumPy.

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("U=")
print(U)
print("S=")
print(np.diag(S))
print("VT=")
print(VT)

reconstructed_matrix= np.dot (U, np.dot(np.diag(S), VT))
print("Reconstructed matrix=")
print(reconstructed_matrix)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonPr
U=
[[-0.59482308  0.7878662  -0.15953794]
 [-0.55395727 -0.54556995 -0.6288758 ]
 [-0.58250909 -0.28569264  0.76096181]]
S=
[[14.28896808  0.          0.        ]
 [ 0.          2.76798539  0.        ]
 [ 0.          0.          0.40453427]]
VT=
[[-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 and the result was successfully obtained. Thus, CO1 was obtained.

Experiment No.: 3

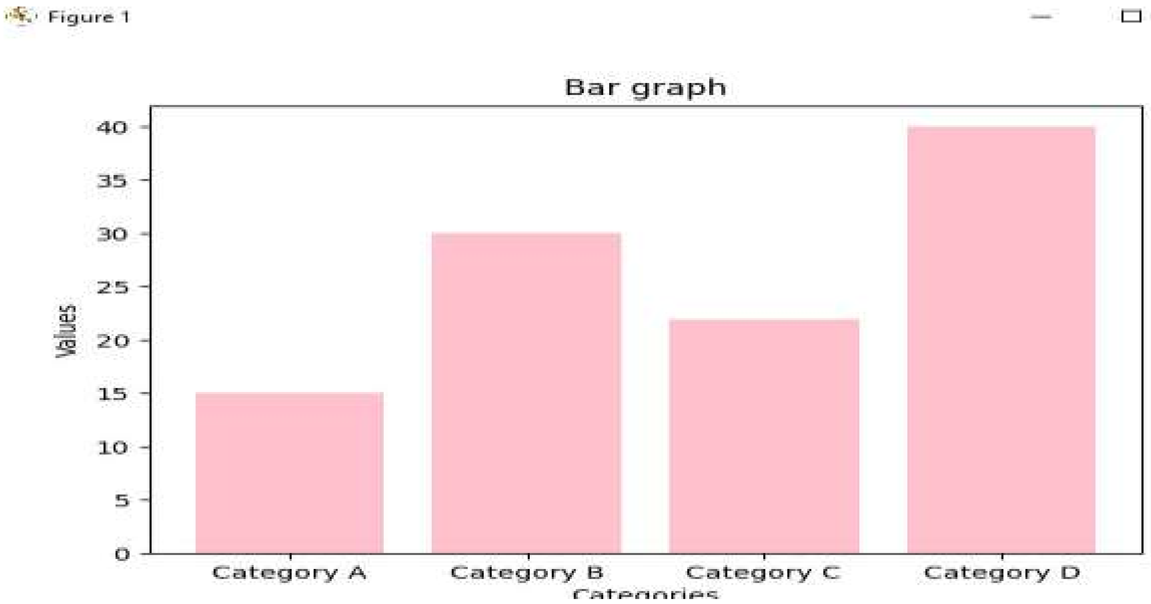
Aim: Program to perform data visualisation using the python library matplotlib.

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

Procedure:

```
import matplotlib.pyplot as plt
categories = ['Category A','Category B','Category C','Category D']
values = [15,30,22,40]
plt.bar(categories,values,color="pink")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.title("Bar graph")
plt.show()
```

Output Screenshot



Result:

The program was executed and the result was successfully obtained. Thus, CO1 was obtained.

Experiment No.: 4

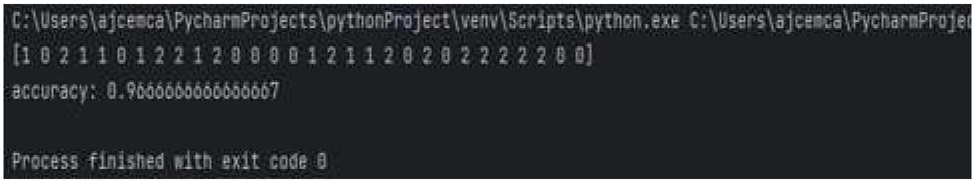
Aim: Program to implement KNN classification using any standard dataset available in the public domain and find the accuracy of the algorithm (Iris Dataset)

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\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjec
[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 and the result was successfully obtained. Thus, CO2 was obtained.

Experiment No.: 5

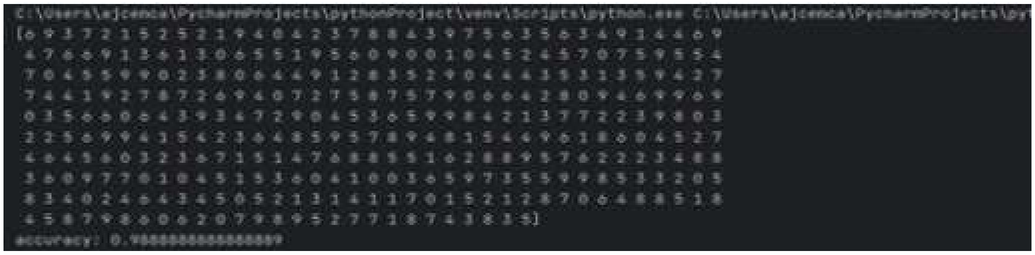
Aim: Program to implement KNN classification using any standard dataset available in the public domain and find the accuracy of the algorithm (Load Digits).

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

Procedure:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_digits
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))
V=knn.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\pyt
[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 5 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 9 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 3 5 4 2 3 6 4 8 5 9 5 7 8 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 8 9 5 2 7 7 1 8 7 4 3 8 3 5]
accuracy: 0.9888888888888889
```

Result:

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

Experiment No.: 6

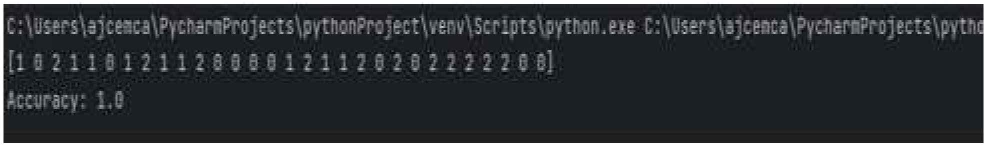
Aim: Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm (Iris Dataset).

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

Procedure:

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.naive_bayes import GaussianNB
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\pytho
[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
```

Result:

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

Experiment No.: 7

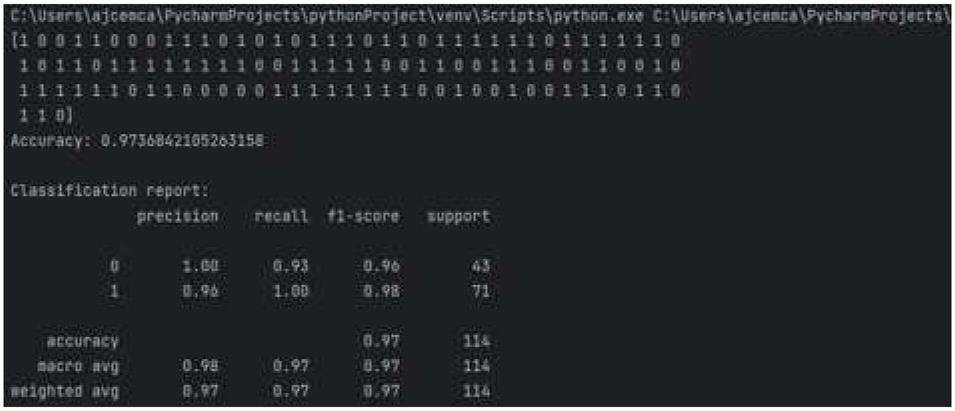
Aim: Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm (Breast Cancer Dataset).

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

Procedure:

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,classification_report
data=load_breast_cancer()
x=data.data
y=data.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)
print()
report=classification_report(y_test,V)
print("Classification report:")
print(report)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\
[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 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 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 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      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

Result:

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

Experiment No.: 8

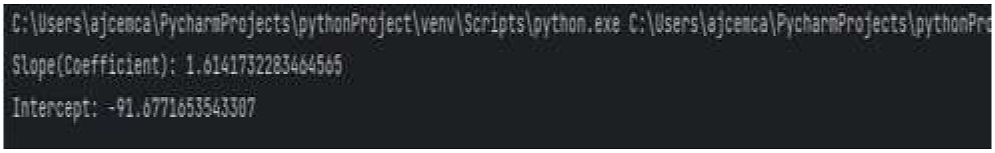
Aim: Given one-dimensional dataset represented with NumPy array. Write a program to calculate slope and intercept.

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

Procedure:

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

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

regressor = LinearRegression()
regressor.fit(x, y)
slope = regressor.coef_[0]
intercept = regressor.intercept_
print(f"Slope(Coefficient): {slope}")
print(f"Intercept: {intercept}")
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonPro
Slope(Coefficient): 1.6141732283464565
Intercept: -91.6771653543307
```

Result:

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

Experiment No.: 9

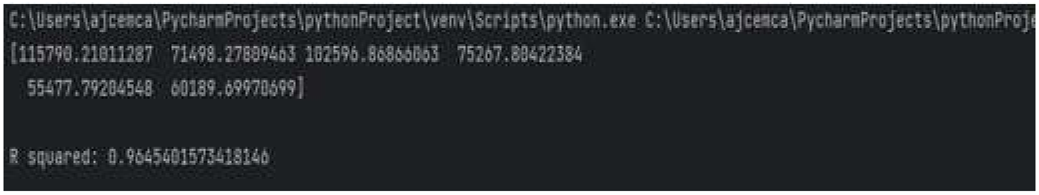
Aim: Program to implement simple linear regression using any standard dataset available in the public domain and find the r2 score.

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

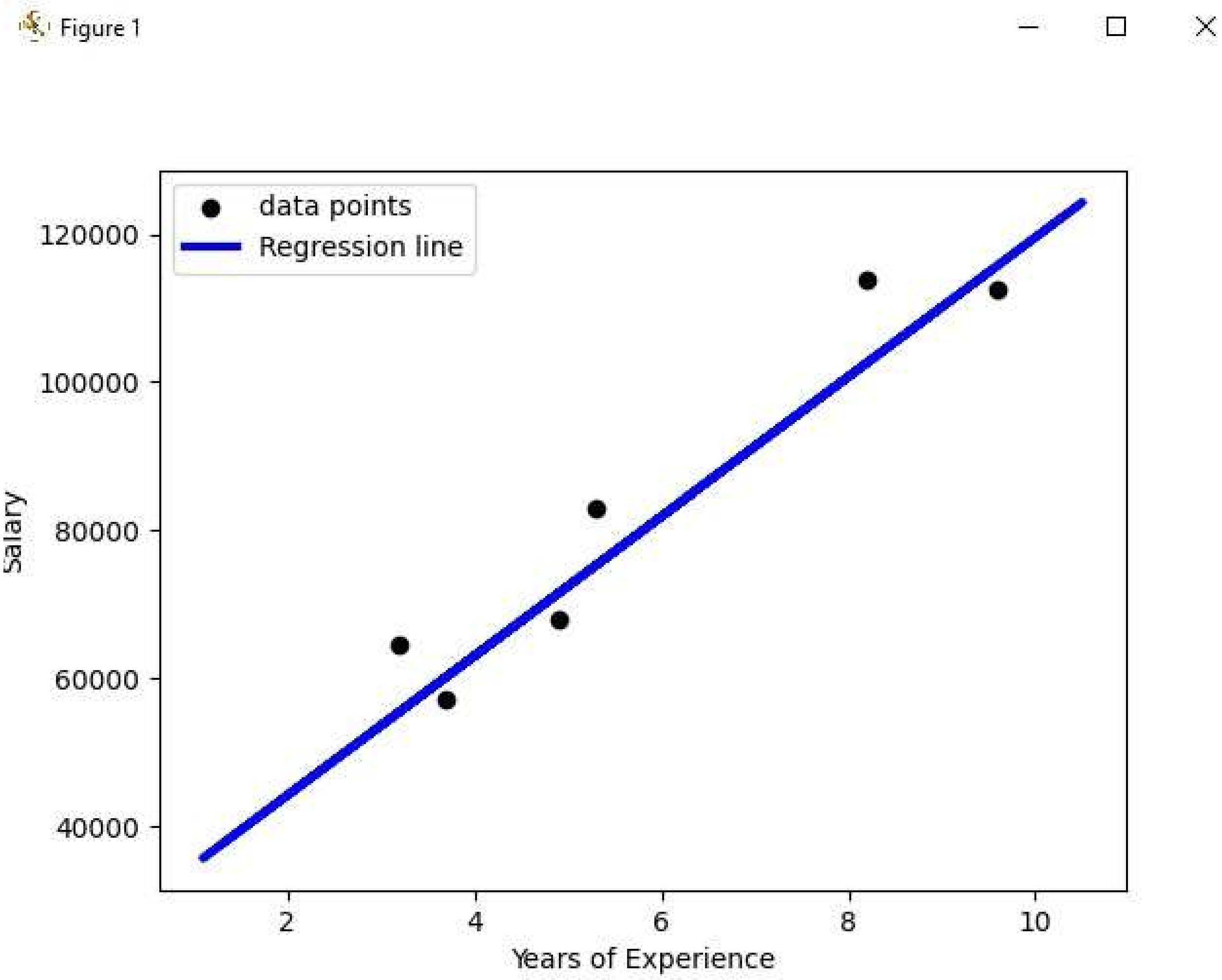
Procedure:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt

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)
regressor = LinearRegression()
regressor.fit(x_train, y_train)
print(regressor.predict(x_test))
V = regressor.predict(x_train)
r2 = r2_score(y_train, V)
print("\nR squared:", r2)
plt.scatter(x_test, y_test, color="black", label="data points")
plt.plot(x_train, V, color="blue", linewidth=3, label="Regression line")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.legend()
plt.show()
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProje
[115790.21011287  71498.27889463 102596.86866063  75267.80422384
  55477.79204548  60189.69970699]

R squared: 0.9645401573418146
```



Result:

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

Experiment No.: 10

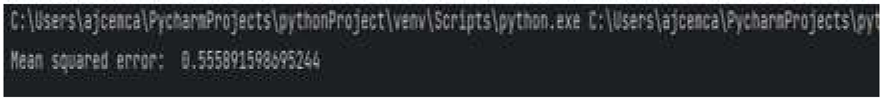
Aim: Program to implement multiple linear regression techniques using any standard dataset available in public domain and evaluate its performance.

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.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
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)
regressor = LinearRegression()
regressor.fit(x_train, y_train)
v = regressor.predict(x_test)
mse = mean_squared_error(y_test, v)
print("Mean squared error: ", mse)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pyt
Mean squared error:  0.555891598695244
```

Result:

The program was executed and the result was successfully obtained. Thus, CO2 was obtained.

Experiment No.: 11

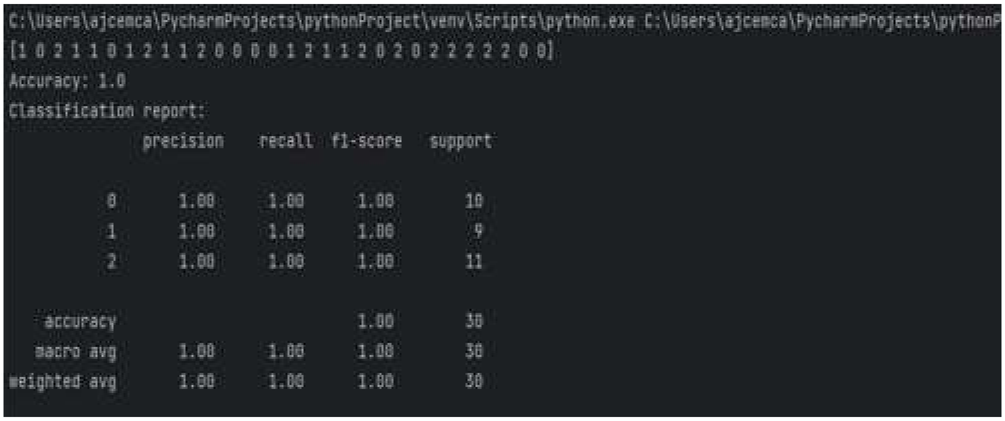
Aim: Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm (Iris Dataset).

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

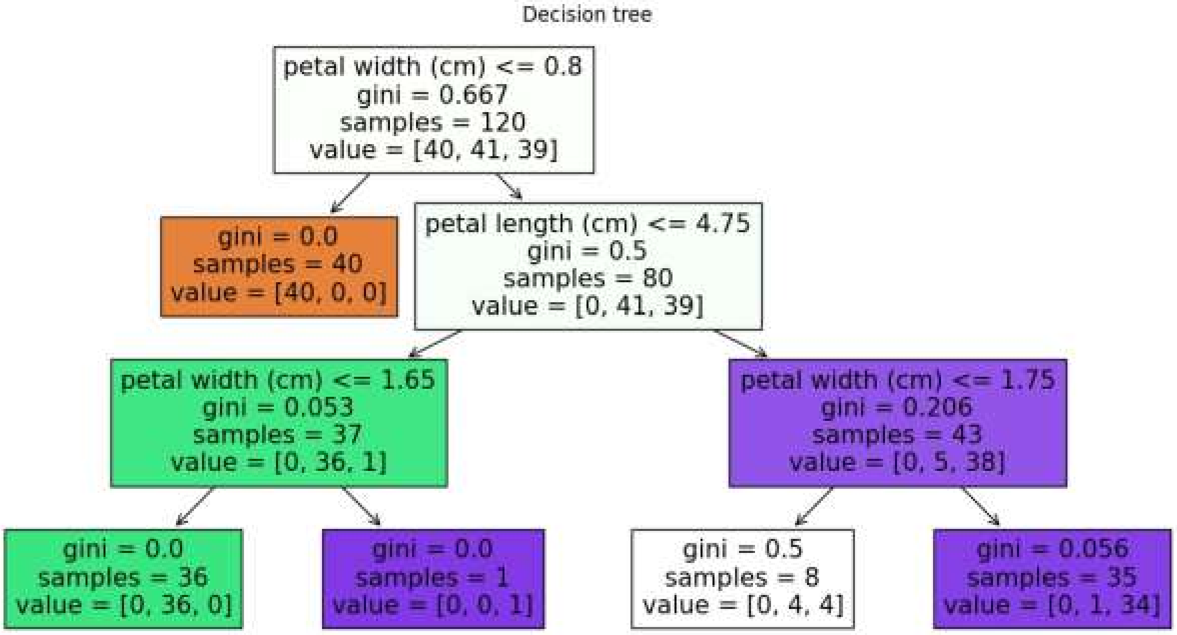
Procedure:

```
from sklearn.datasets import load_iris
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
import matplotlib.pyplot as plt
iris=load_iris()
x=iris.data
y=iris.target
max_depth=3
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
clf=DecisionTreeClassifier(max_depth=max_depth)
clf.fit(x_train,y_train)
plt.figure(figsize=(15,10))
tree.plot_tree(clf,filled=True,feature_names=iris.feature_names)
plt.title("Decision tree")
plt.show()
print(clf.predict(x_test))
V=clf.predict(x_test)
result=accuracy_score(y_test, V)
report=classification_report(y_test,V)
print("Accuracy:",result)
print("Classification report:\n",report)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonP
[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        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```



Result:

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

Experiment No.: 12

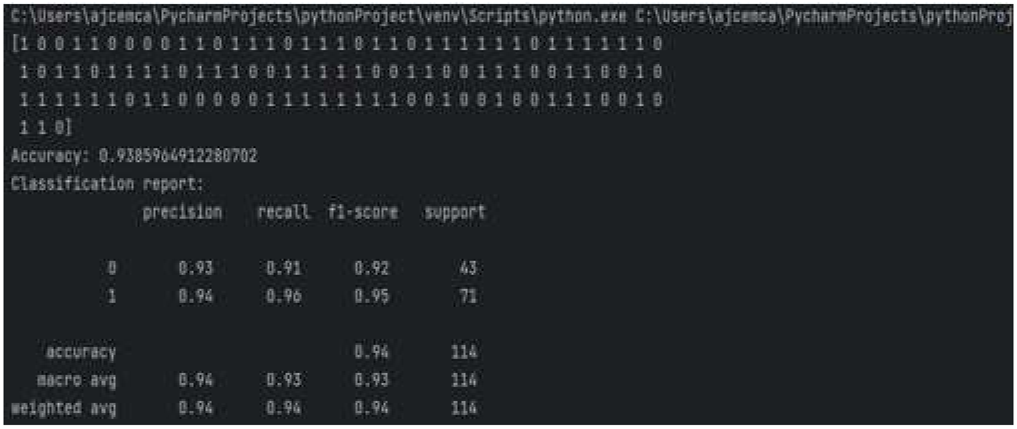
Aim: Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm (Breast Cancer Dataset)

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

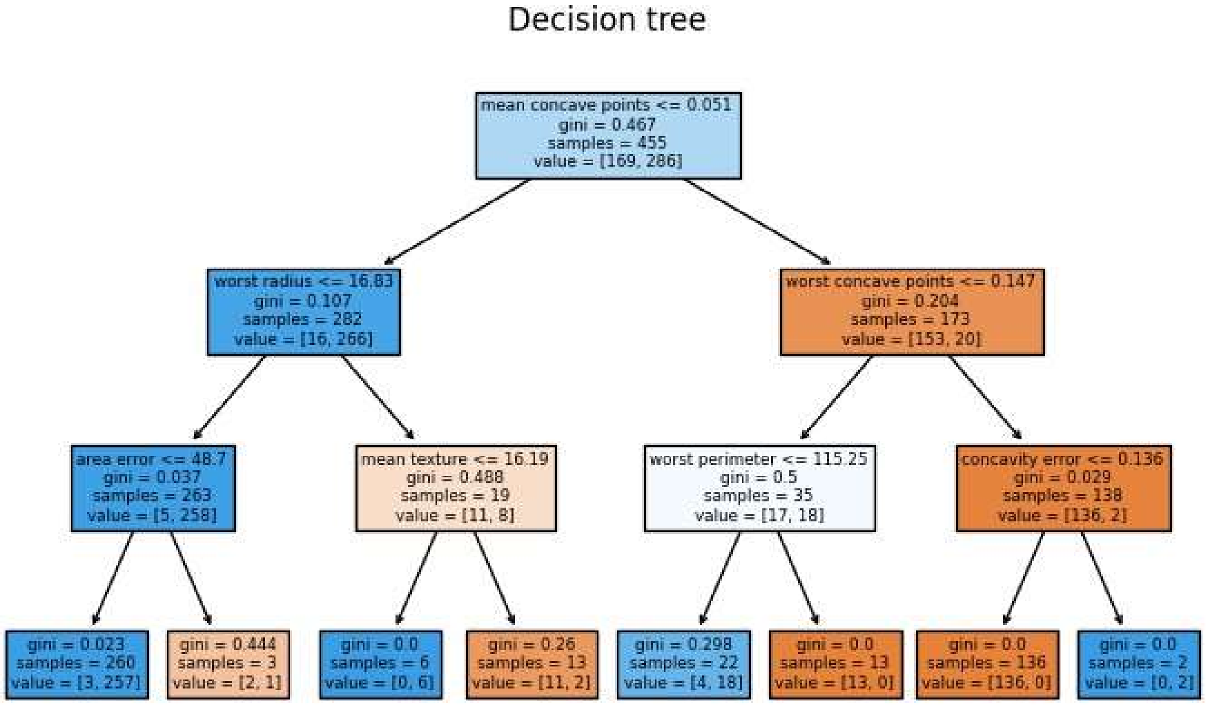
Procedure:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
from sklearn import tree
import matplotlib.pyplot as plt
data=load_breast_cancer()
x=data.data
y=data.target
max_depth=3
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
clf=DecisionTreeClassifier(max_depth=max_depth)
clf.fit(x_train,y_train)
plt.figure(figsize=(15,10))
tree.plot_tree(clf,filled=True,feature_names=data.feature_names)
plt.title("Decision tree")
plt.show()
print(clf.predict(x_test))
V=clf.predict(x_test)
result=accuracy_score(y_test, V)
report=classification_report(y_test,V)
print("Accuracy:",result)
print("Classification report:\n",report)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonProj
[1 0 0 1 1 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0
 1 0 1 1 0 1 1 1 1 0 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 0 1 1 0 0 0 0 0 1 1 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: 0.9385964912280702
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.91      0.92        43
           1       0.94      0.96      0.95        71

    accuracy                           0.94       114
   macro avg       0.94      0.93      0.93       114
weighted avg       0.94      0.94      0.94       114
```



Result:

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

Experiment No.: 13

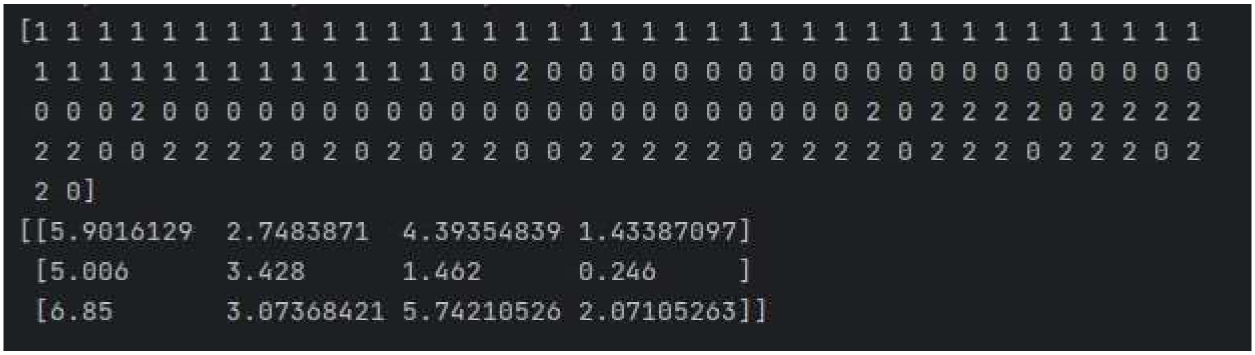
Aim: Program to implement k-means clustering technique using any standard dataset available in the public domain (Iris Dataset)

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

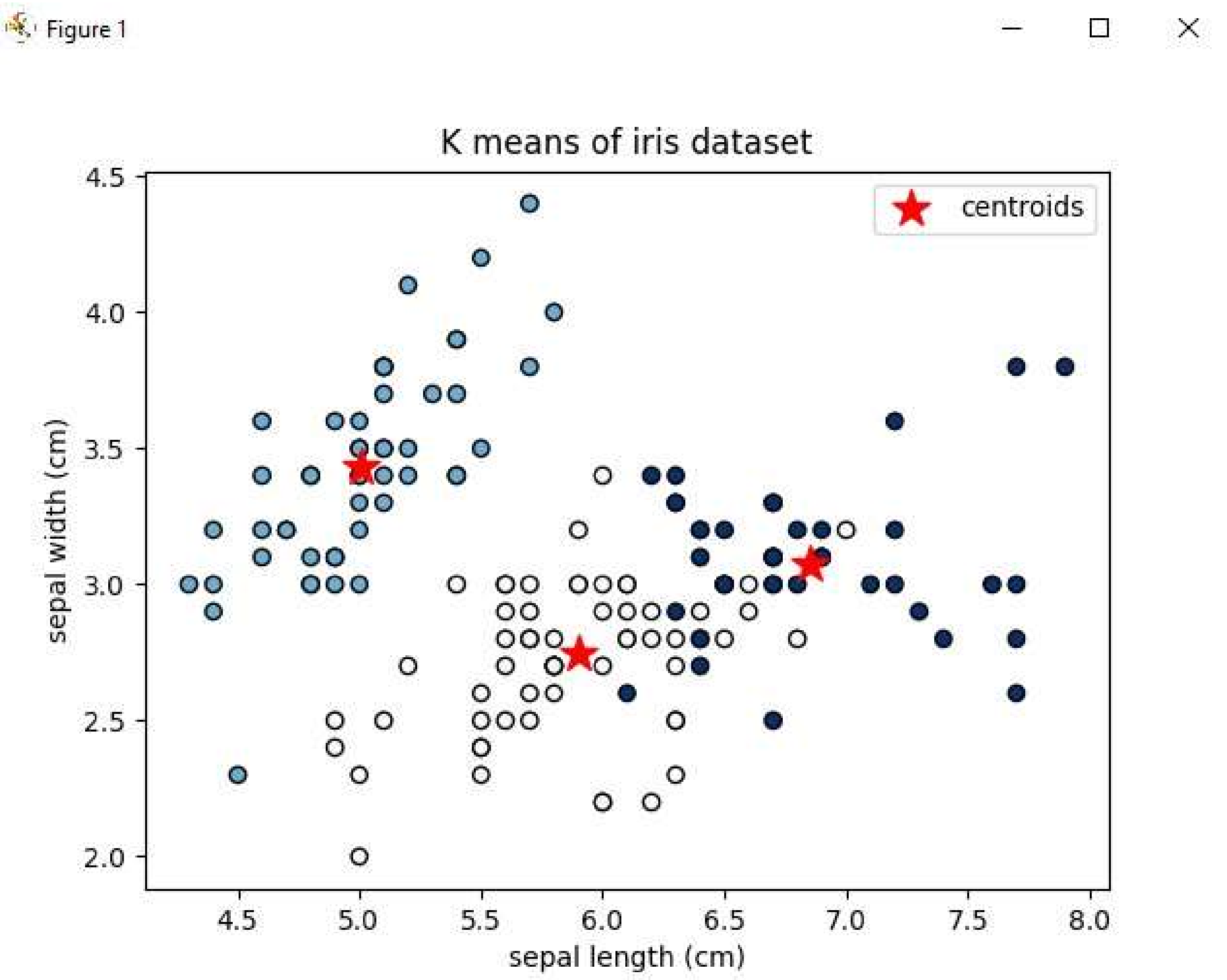
Procedure:

```
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="Blues", 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("K means 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 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 and the result was successfully obtained. Thus, CO3 was obtained.

Experiment No.: 14

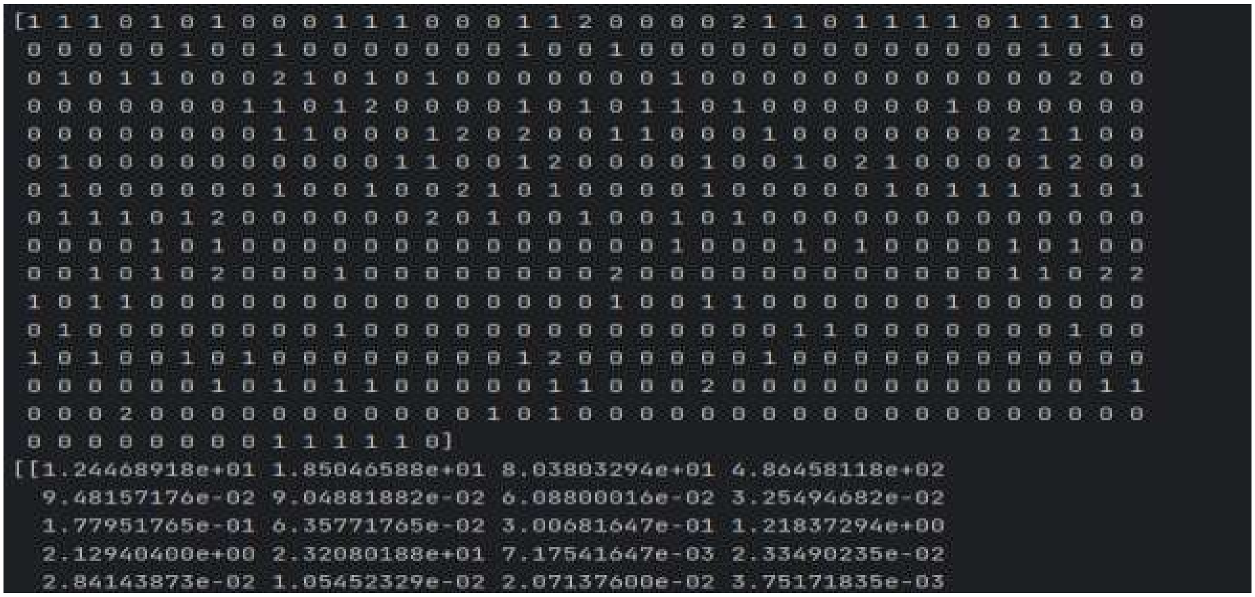
Aim: Program to implement k-means clustering technique using any standard dataset available in the public domain (Breast Cancer Dataset)

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

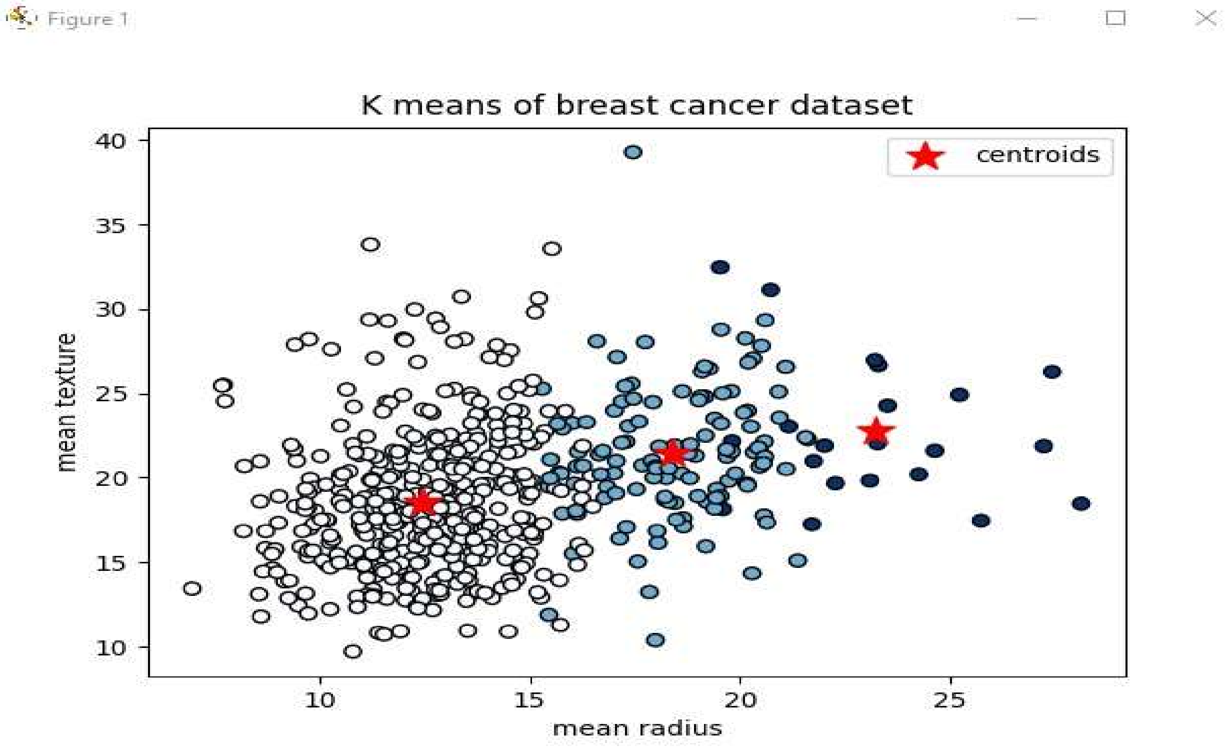
Procedure:

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

data = load_breast_cancer()
x = data.data
y = data.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="Blues", marker="o", edgecolors="black")
plt.scatter(centroids[:, 0], centroids[:, 1], marker="*", s=200, c="red", label="centroids")
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
plt.title("K means 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 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 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 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 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 0 1 1 0 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 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 0 1 1
 0 0 0 2 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 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
```



Result:

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

Experiment No.: 15

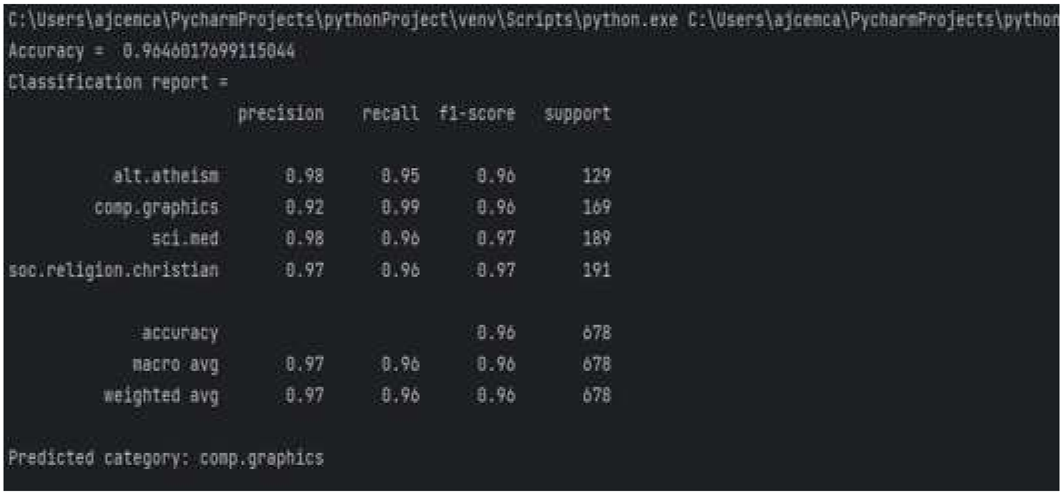
Aim: Program to implement text classification using support vector machine.

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)
# print(x_train_tfidf)
y_train = twenty_train.target
x_train, x_test, y_train, y_test = train_test_split(x_train_tfidf, y_train, test_size=0.3,
random_state=42)
svm_classifier = SVC(kernel='linear', random_state=42)
svm_classifier.fit(x_train, y_train)
prediction = svm_classifier.predict(x_test)
accuracy = accuracy_score(y_test, prediction)
report = classification_report(y_test,prediction,target_names=twenty_train.target_names)
print("Accuracy = ",accuracy)
print("Classification report =")
print(report)
new_data=[
    "I have a question about computer graphics"
]
x_new=vectorizer.transform(new_data)
newprediction=svm_classifier.predict(x_new)
predicted_category=twenty_train.target_names[newprediction[0]]
print("Predicted category:", predicted_category)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\python
Accuracy =  0.9646017699115044
Classification report =
                        precision    recall  f1-score   support

           alt.atheism       0.98      0.95      0.96       129
         comp.graphics       0.92      0.99      0.96       169
               sci.med       0.98      0.96      0.97       189
soc.religion.christian       0.97      0.96      0.97       191

              accuracy                           0.96       678
             macro avg       0.97      0.96      0.96       678
          weighted avg       0.97      0.96      0.96       678

Predicted category: comp.graphics
```

Result:

The program was executed and the result was successfully obtained. Thus, CO3 was obtained.

Experiment No.: 16

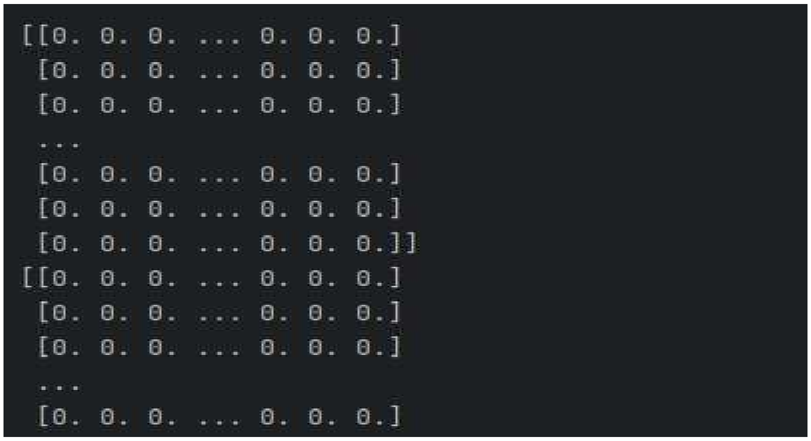
Aim: Program on artificial neural network to classify images from any standard dataset in the public domain using Keras framework.

CO4: Implement convolutional neural network algorithm using Keras framework.

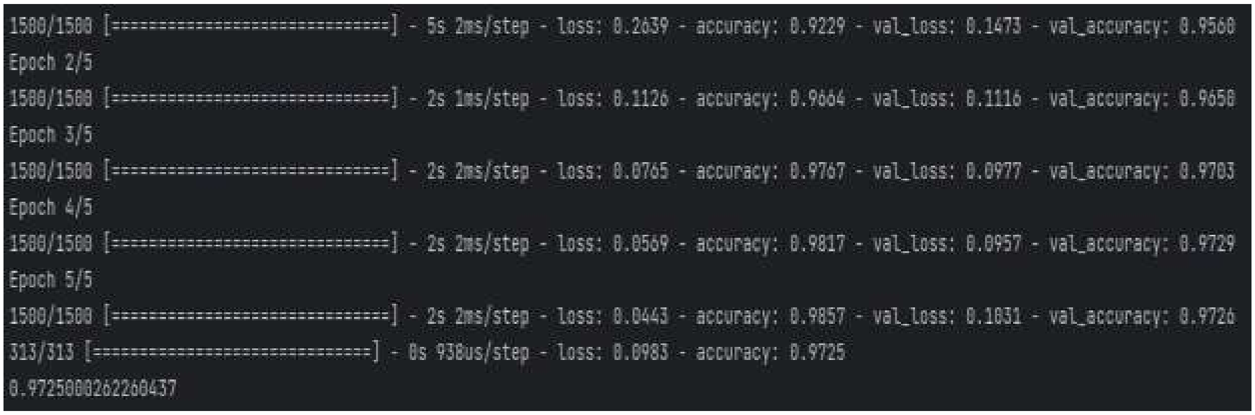
Procedure:

```
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_train)
# 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



```
[[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. 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. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
```



```
1500/1500 [==============================] - 5s 2ms/step - loss: 0.2639 - accuracy: 0.9229 - val_loss: 0.1473 - val_accuracy: 0.9560
Epoch 2/5
1500/1500 [==============================] - 2s 1ms/step - loss: 0.1126 - accuracy: 0.9664 - val_loss: 0.1116 - val_accuracy: 0.9650
Epoch 3/5
1500/1500 [==============================] - 2s 2ms/step - loss: 0.0765 - accuracy: 0.9767 - val_loss: 0.0977 - val_accuracy: 0.9703
Epoch 4/5
1500/1500 [==============================] - 2s 2ms/step - loss: 0.0569 - accuracy: 0.9817 - val_loss: 0.0957 - val_accuracy: 0.9729
Epoch 5/5
1500/1500 [==============================] - 2s 2ms/step - loss: 0.0443 - accuracy: 0.9857 - val_loss: 0.1031 - val_accuracy: 0.9726
313/313 [==============================] - 0s 938us/step - loss: 0.0983 - accuracy: 0.9725
0.9725000262260437
```

Result:

The program was executed and the result was successfully obtained. Thus, CO4 was obtained.

Experiment No.: 17

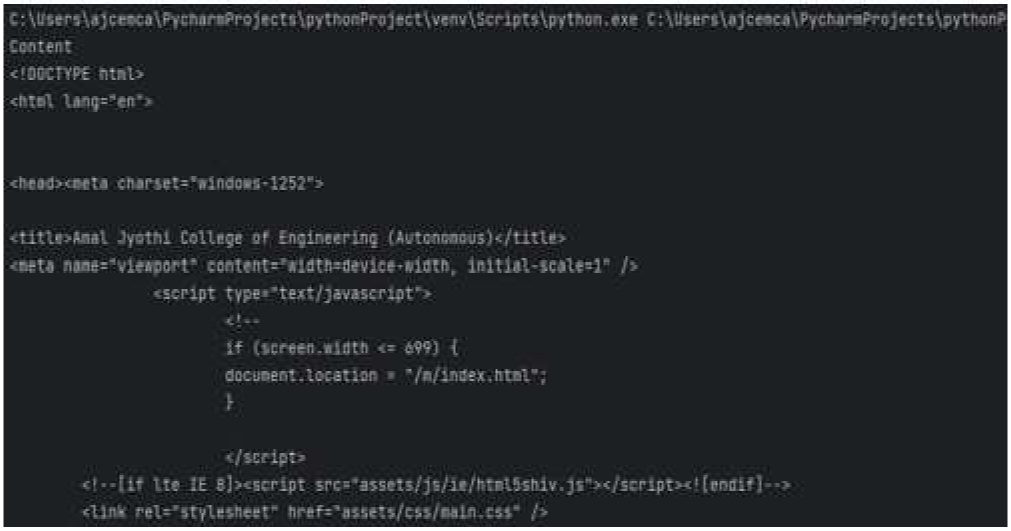
Aim: Program to implement a simple web crawler and scrapping web pages.

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 = 'http://ajce.in'
simple_scraper(url_to_scrap)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\pythonP
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" />
```

Result:

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.

Experiment No.: 18

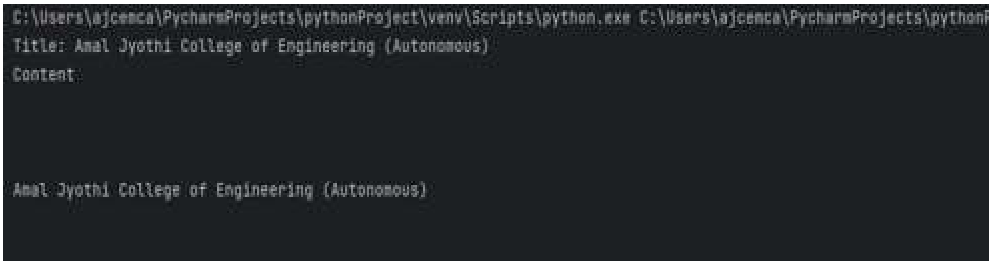
Aim: Program to implement a simple web crawler and parse the content using BeautifulSoup.

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

Procedure:

```
import requests
from bs4 import BeautifulSoup
def simple_scraper_with_bs(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 = 'http://ajce.in'
simple_scraper_with_bs(url_to_scrap)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\PycharmProjects\python
Title: Amal Jyothi College of Engineering (Autonomous)
Content



Amal Jyothi College of Engineering (Autonomous)
```

Result:

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.

Experiment No.: 19

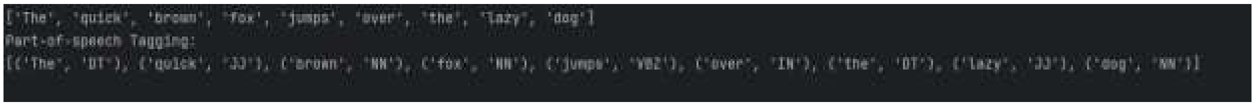
Aim: Implement problems on natural language processing - Part of Speech tagging, N-gram & smoothening and Chunking using NLTK

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

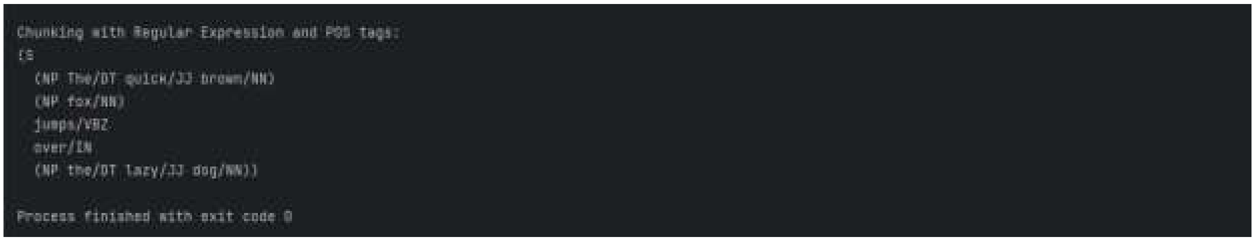
Procedure:

```
import nltk

nltk.download('punkt')
nltk.download('brown')
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("\nN-gram Analysis(Bigram with Smoothening)")
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("\nChunking with Regular Expression 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')]
```



```
Chunking with Regular Expression and POS tags:
(S
  (NP The/DT quick/JJ brown/NN)
  (NP fox/NN)
  jumps/VBZ
  over/IN
  (NP the/DT lazy/JJ dog/NN))

Process finished with exit code 0
```

Result:

The program was executed and the result was successfully obtained. Thus, CO5 was obtained.