Experiment No.: 1

Aim

Program to perform matrix operations. Use numpy as the python library and perform the operation using built in functions.

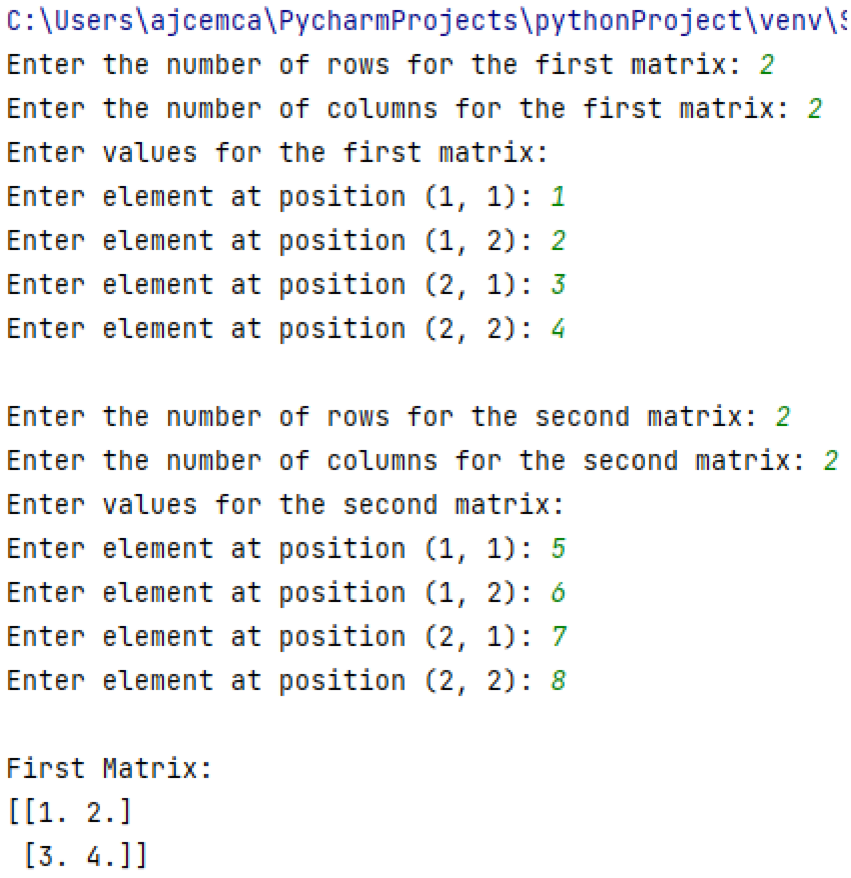
CO1

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

Procedure

```
import numpy as np
rows1 = int(input("Enter the number of rows for the first matrix: "))
cols1 = int(input("Enter the number of columns for the first matrix: "))
matrix1 = np.empty((rows1, cols1), dtype=float)
print("Enter values for the first matrix:")
for i in range(rows1):
    for j in range(cols1):
        matrix1[i][j] = float(input(f"Enter element at position ({i + 1}, {j + 1}): "))
rows2 = int(input("\nEnter the number of rows for the second matrix: "))
cols2 = int(input("Enter the number of columns for the second matrix: "))
matrix2 = np.empty((rows2, cols2), dtype=float)
print("Enter values for the second matrix:")
for i in range(rows2):
    for j in range(cols2):
        matrix2[i][j] = float(input(f"Enter element at position ({i + 1}, {j + 1}): "))
print("\nFirst Matrix:")
print(matrix1)
print("\nSecond Matrix:")
print(matrix2)
matrix1=np.array([[1,2],[3,4]])
matrix2=np.array([[5,6],[7,8]])
print(matrix1)
print(matrix2)
sum1=np.add(matrix1,matrix2)
sub1=np.subtract(matrix1,matrix2)
mul1=np.multiply(matrix1,matrix2)
div1=np.divide(matrix1,matrix2)
print("\n Sum:", sum1)
print("\n Subtraction:", sub1)
print("\n Multiplication:", mul1)
print("\n Division:", div1)
```

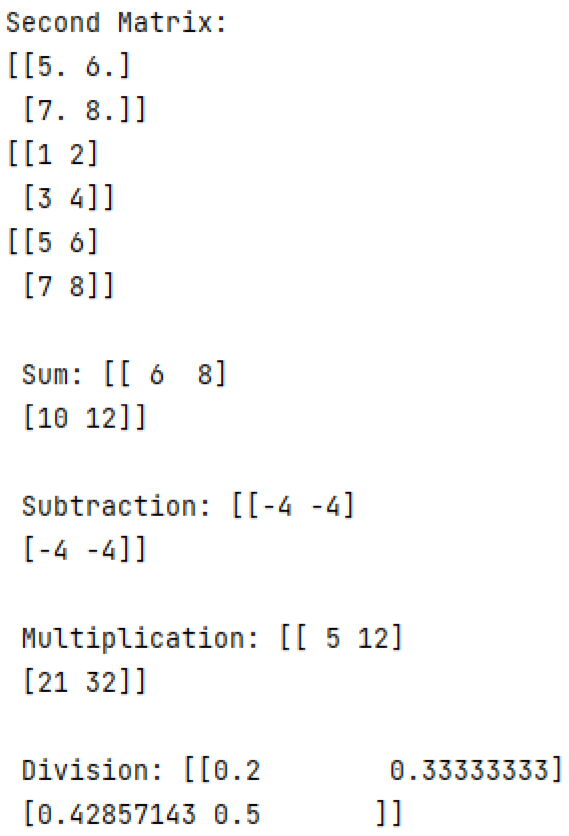
Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\
Enter the number of rows for the first matrix: 2
Enter the number of columns for the first matrix: 2
Enter values for the first matrix:
Enter element at position (1, 1): 1
Enter element at position (1, 2): 2
Enter element at position (2, 1): 3
Enter element at position (2, 2): 4

Enter the number of rows for the second matrix: 2
Enter the number of columns for the second matrix: 2
Enter values for the second matrix:
Enter element at position (1, 1): 5
Enter element at position (1, 2): 6
Enter element at position (2, 1): 7
Enter element at position (2, 2): 8

First Matrix:
[[1. 2.]
 [3. 4.]]
```



```
Second Matrix:
[[5. 6.]
 [7. 8.]]
[[1 2]
 [3 4]]
[[5 6]
 [7 8]]

 Sum: [[ 6  8]
 [10 12]]

 Subtraction: [[-4 -4]
 [-4 -4]]

 Multiplication: [[ 5 12]
 [21 32]]

 Division: [[0.2        0.33333333]
 [0.42857143 0.5       ]]
```

Result

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

Experiment No.:2

Aim

Program to perform single value decomposition using numpy.

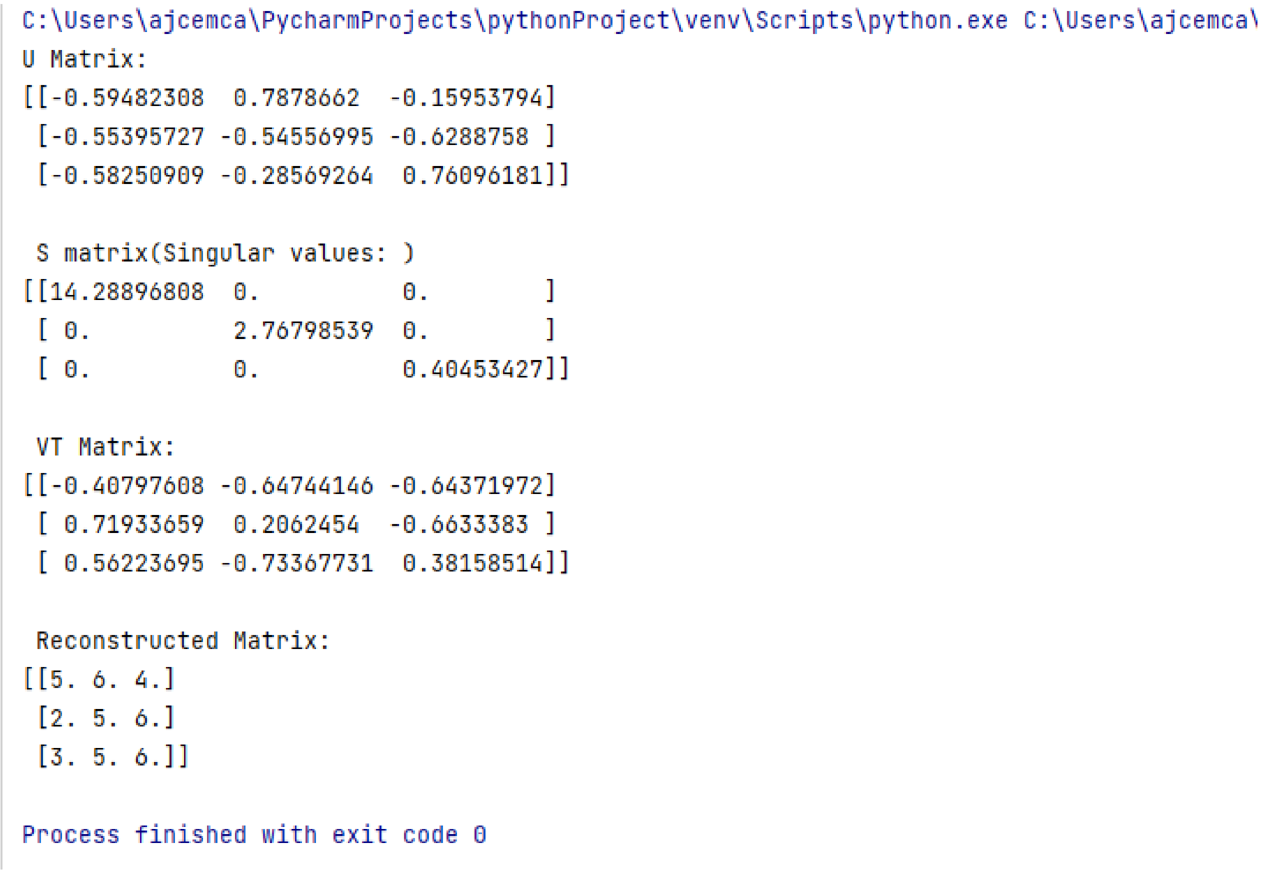
CO1

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 Matrix: ")
print(U)
print("\n S matrix(Singular values: )")
print(np.diag(S))
print("\n VT Matrix: ")
print(VT)
reconstructed_matrix = np.dot(U, np.dot(np.diag(S),VT))
print("\n Reconstructed Matrix: ")
print(reconstructed_matrix)
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\ajcemca\
U Matrix:
[[-0.59482308  0.7878662  -0.15953794]
 [-0.55395727 -0.54556995 -0.6288758 ]
 [-0.58250909 -0.28569264  0.76096181]]

 S matrix(Singular values: )
[[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 was attained.

Experiment No.: 3

Aim

Program to perform data visualization using python library matplotlib

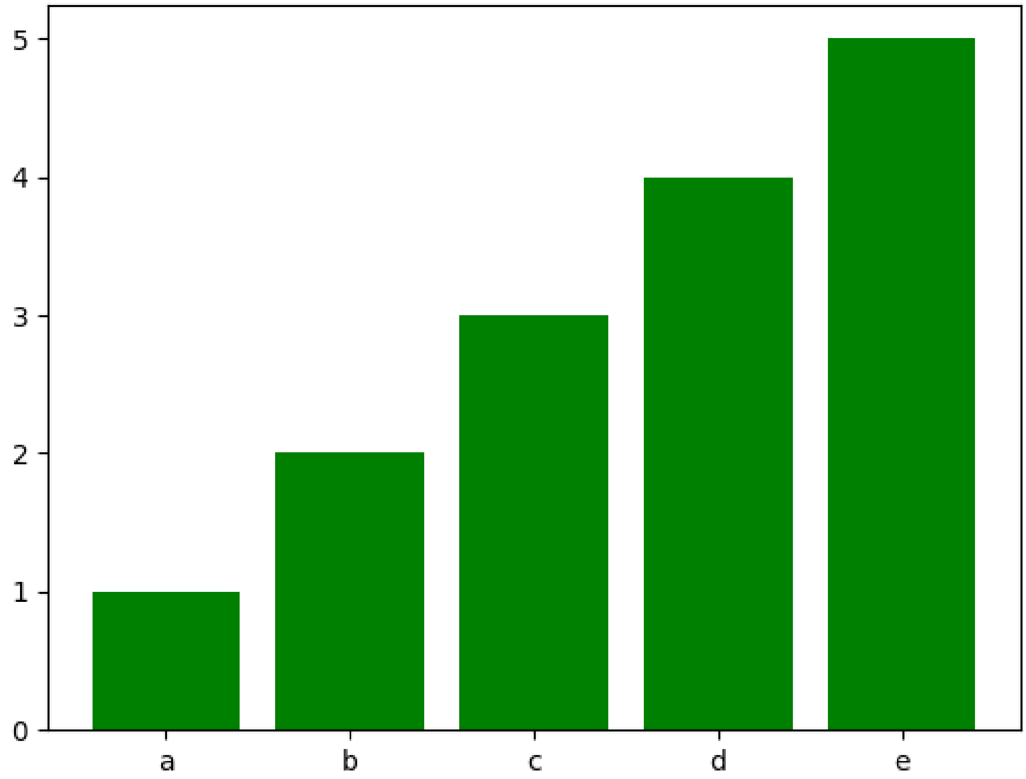
CO1

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

Procedure

```
import matplotlib as plt
import matplotlib.pyplot as plt
category=['a','b','c','d','e']
values=[1,2,3,4,5]
plt.bar(category,values,color='green')
plt.show()
```

Output Screenshot



Result

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

Experiment No.: 4

Aim

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

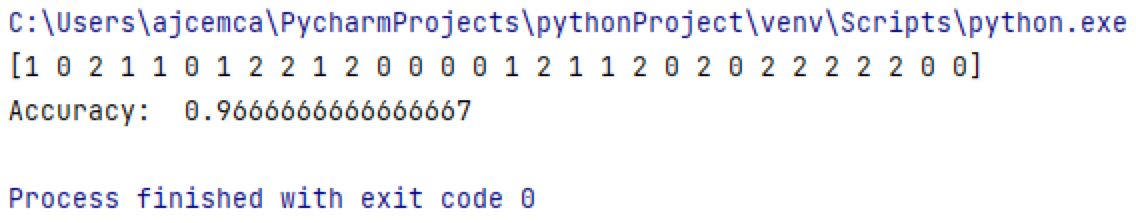
CO2

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 #features
y = iris.target #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
[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 was attained.

Experiment No.: 5

Aim

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

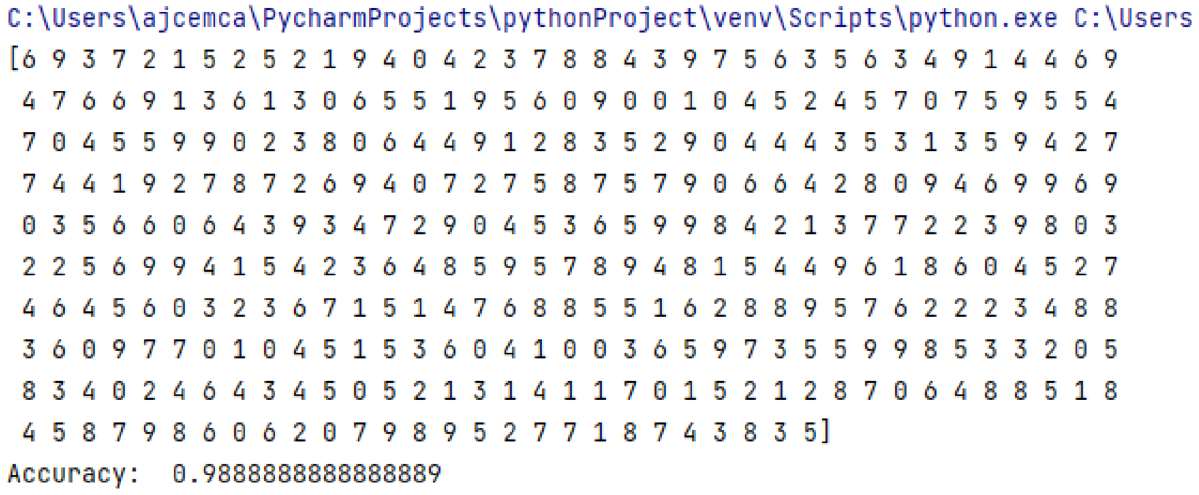
CO2

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
[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 1 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 successfully and the output was obtained. Thus CO2 was 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 algorithm. (Iris Dataset)

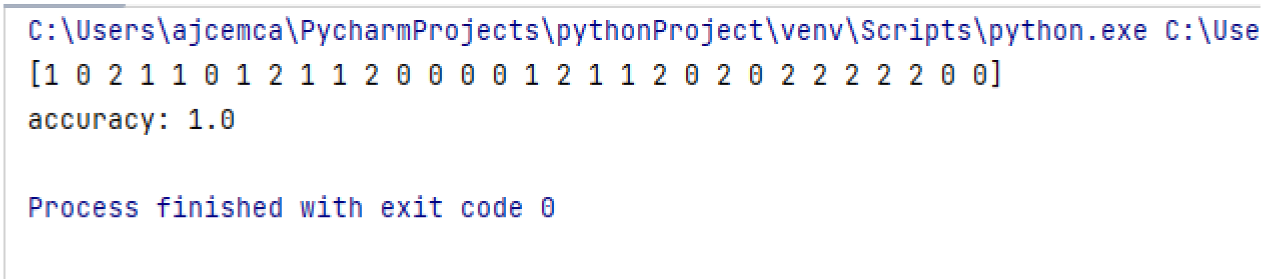
CO2

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:\Use
[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

Process finished with exit code 0
```

Result

The program was executed successfully and the output was obtained. Thus CO2 was 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 algorithm. (Breast Cancer Dataset)

CO2

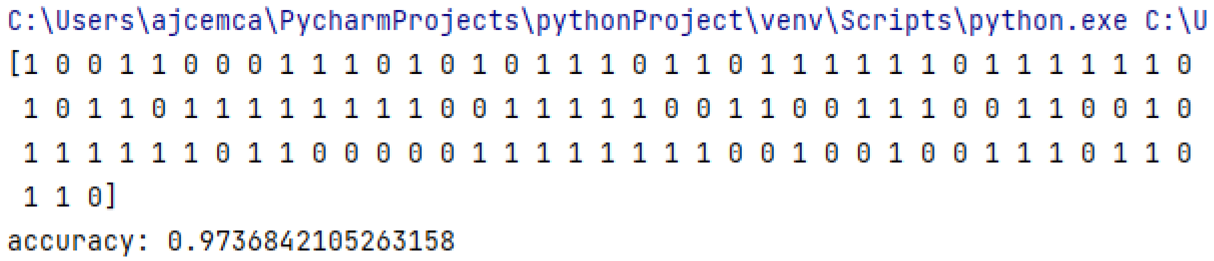
Use different packages and frameworks to implement regression and classification algorithms.

Procedure

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from  sklearn.metrics import accuracy_score

iris=load_breast_cancer()
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:\U
[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
```

Result

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

Experiment No.: 8

Aim

Give one dimensional dataset represented with numpy array. Write a program to calculate slope and intercept

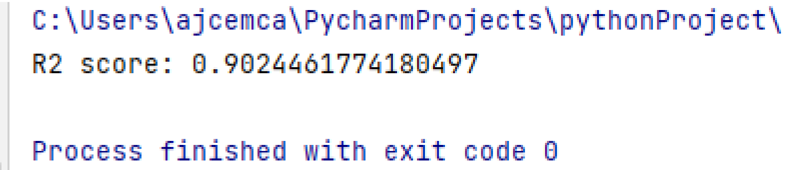
CO2

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

Procedure

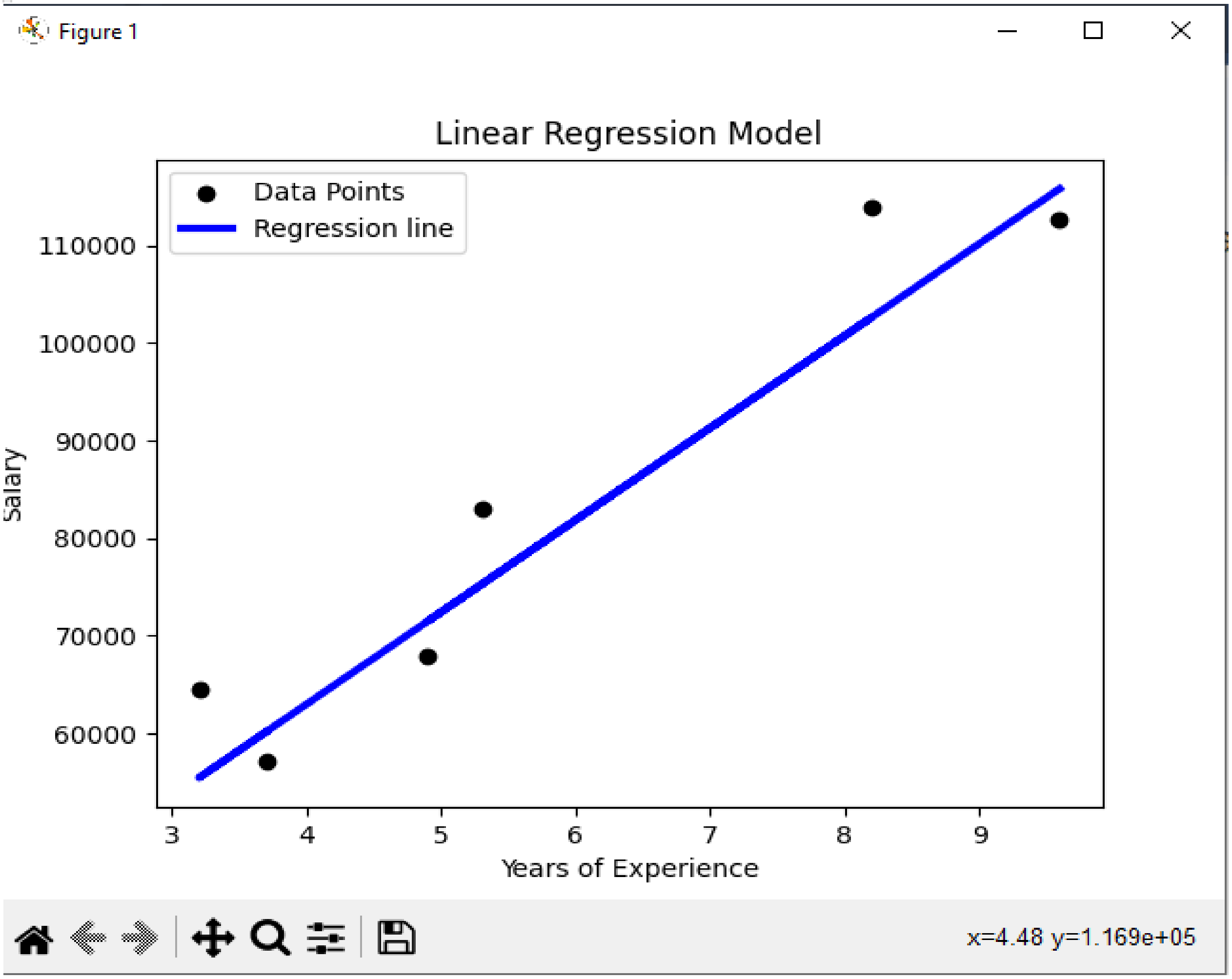
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
data = pd.read_csv('Salary_Data.csv')
x = data['YearsExperience'].values.reshape(-1, 1)
y = data['Salary'].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:", r2)
plt.scatter(x_test, y_test, color='black', label='Data Points')
plt.plot(x_test, y_pred, color='blue', linewidth=3, label='Regression line')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.title('Linear Regression Model')
plt.show()
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\
R2 score: 0.9024461774180497

Process finished with exit code 0
```



Result

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

Experiment No.: 9

Aim

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

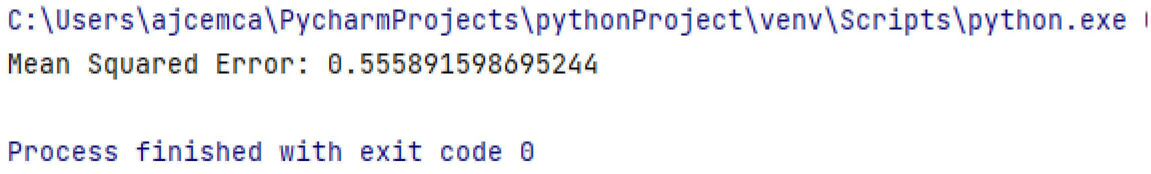
CO2

Use different packages and frameworks to implement regression and classification algorithms

Procedure

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
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)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe
Mean Squared Error: 0.555891598695244

Process finished with exit code 0
```

Result

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

Experiment No.: 10

Aim

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

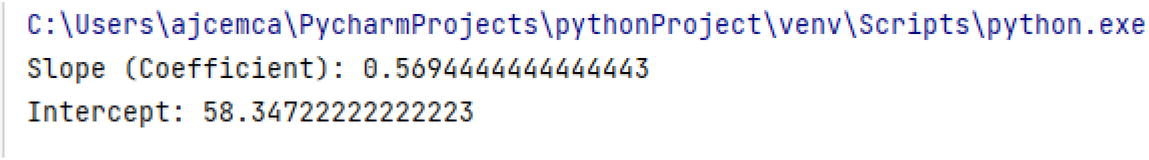
CO2

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

Procedure

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
y_values=np.array([17, 27, 15, 24, 39, 44, 30, 48, 19, 47]).reshape(-1,1)
x_values=np.array([64, 75, 68, 73, 78, 82, 76, 85, 71, 88])
model=LinearRegression()
model.fit(y_values, x_values)
slope=model.coef_[0]
intercept=model.intercept_
print(f"Slope (Coefficient): {slope}")
print(f"Intercept: {intercept}")
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe
Slope (Coefficient): 0.5694444444444443
Intercept: 58.34722222222223
```

Result

The program was executed successfully and the output was obtained. Thus CO2 was 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. (Iris Dataset)

CO3

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

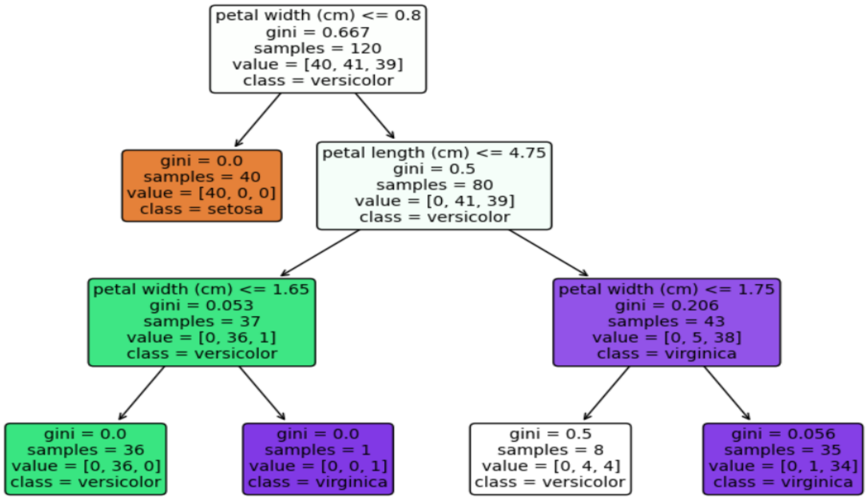
Procedure

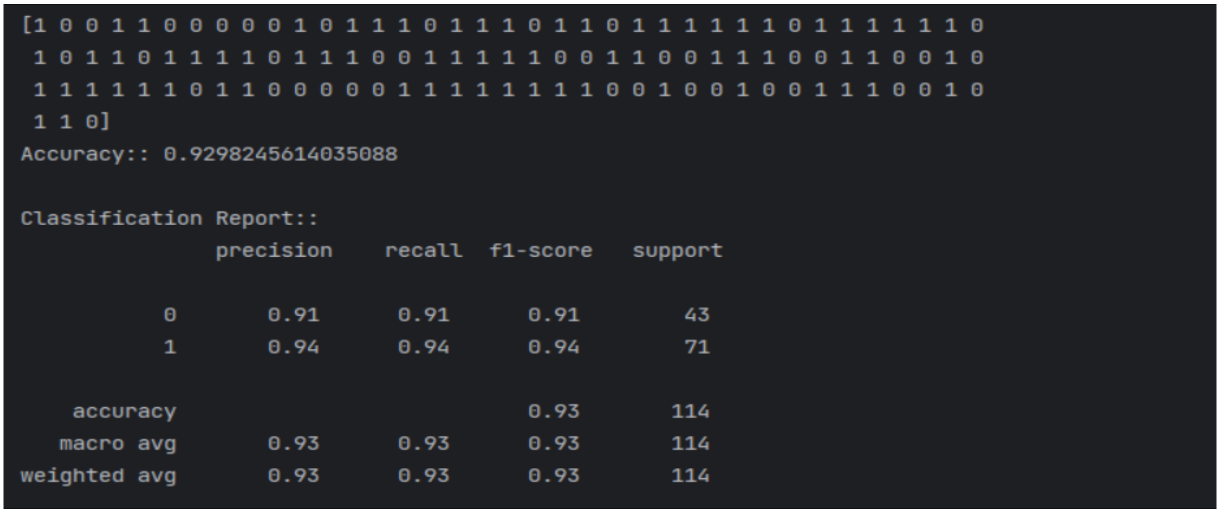
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score,classification_report
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
iris=load_iris()
x=iris.data
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))
v=dt.predict(x_test)
report=classification_report(y_test,v)
result=accuracy_score(y_test,v)
print("Accuracy::",result)
print("\nClassification Report::\n",report)
plt.figure("DECISION TREE",figsize=(10, 10))
```

plot_tree(dt,filled=True,feature_names=iris.feature_names,class_names=iris.target_names, rounded=True)

plt.show()

Output Screenshot:





```
[1 0 0 1 1 0 0 0 0 0 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 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       0.91      0.91      0.91        43
           1       0.94      0.94      0.94        71

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

Result

The program was executed successfully and the output was obtained. Thus CO3 was 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. (Breast Cancer Dataset)

CO3

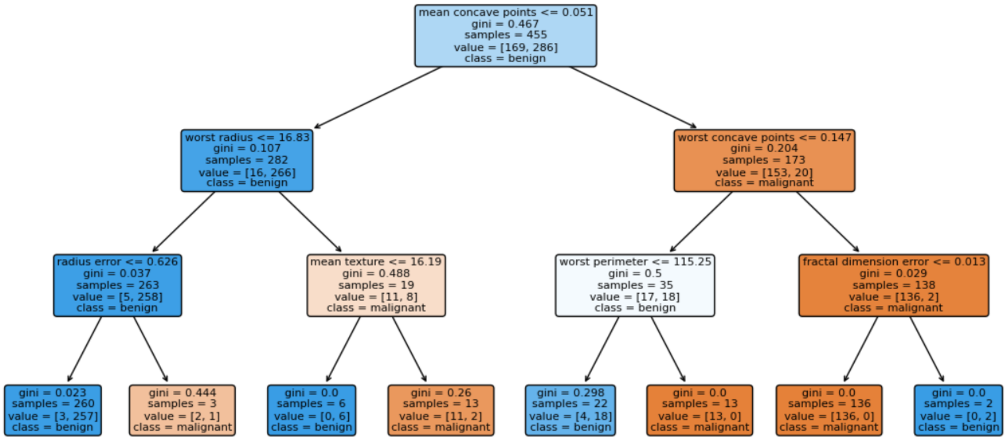
Use different packages and frameworks to implement text classification using SVM and clustering using k-means.

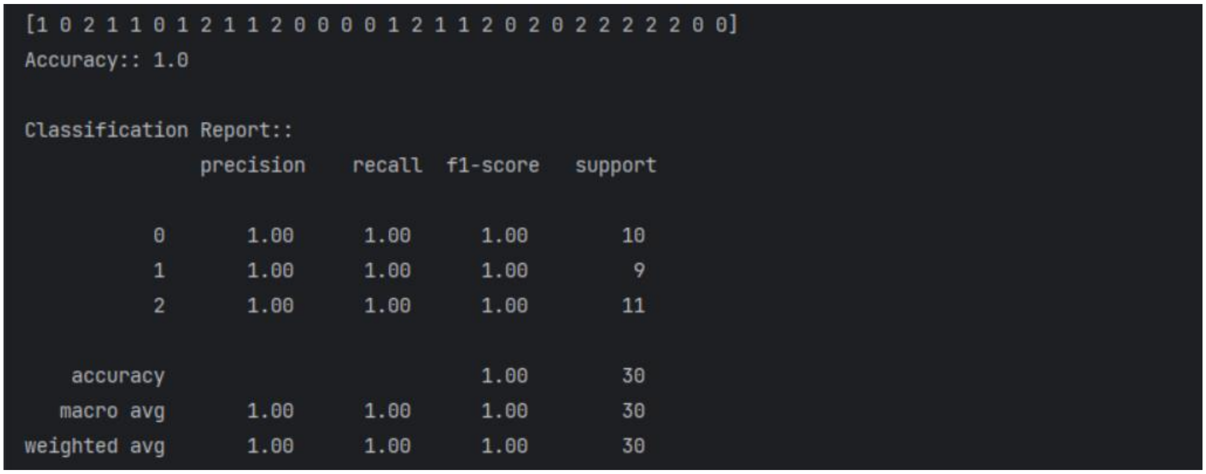
Procedure

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import accuracy_score, classification_report
from sklearn.tree import plot_tree
import matplotlib.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=3)
dt.fit(x_train, y_train)
v = dt.predict(x_test)
report = classification_report(y_test, v)
result = accuracy_score(y_test, v)
print("Accuracy:", result)
print("\nClassification Report:\n", report)
plt.figure("DECISION TREE",figsize=(20, 10))
plot_tree(dt, filled=True, feature_names=bc.feature_names, class_names=bc.target_names, rounded=True)
```

```
plt.show()
```

Output Screenshot





```
[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 successfully and the output was obtained. Thus CO3 was attained.

Experiment No:13

Aim

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

CO3

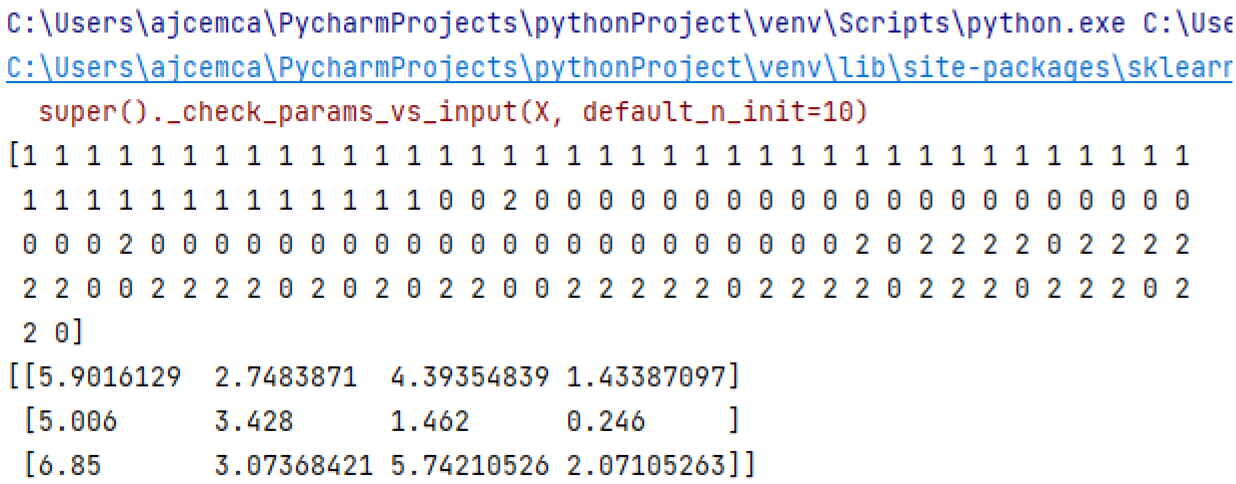
Use different packages and frameworks to implement text classification using SVM and clustering using k-means.

Procedure

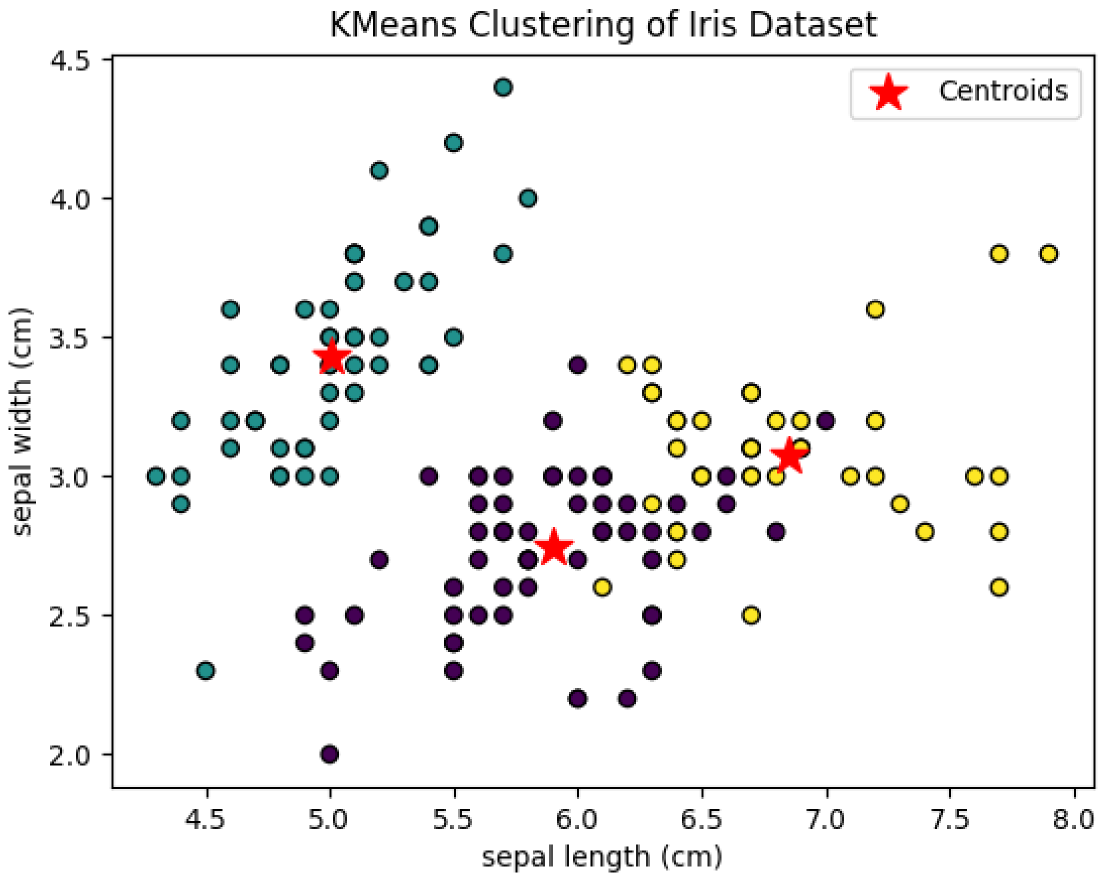
```
from sklearn.datasets import load_iris

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='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 Clustering of Iris Dataset')
plt.legend()
plt.show()
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Use
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\lib\site-packages\sklearr
  super()._check_params_vs_input(X, default_n_init=10)
[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 successfully and the output was obtained. Thus CO3 was attained.

Experiment No:14

Aim

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

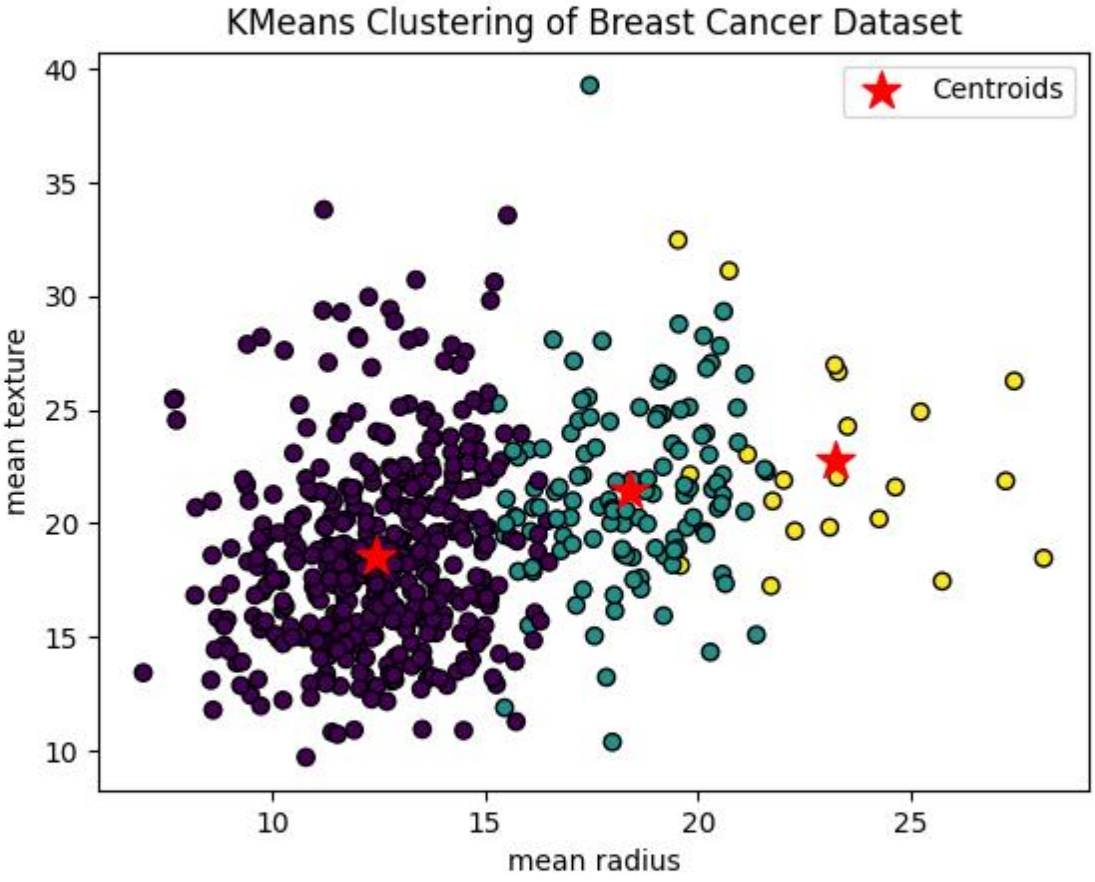
CO3

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

Procedure

```
 from sklearn.cluster import KMeans
from sklearn.datasets import load_breast_cancer
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 Clustering of Breast Cancer Dataset')
plt.legend()
plt.show()
```

Output Screenshot



Result

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

Experiment No.: 15

Aim

Program to implement test classification using Support Vector Machine.

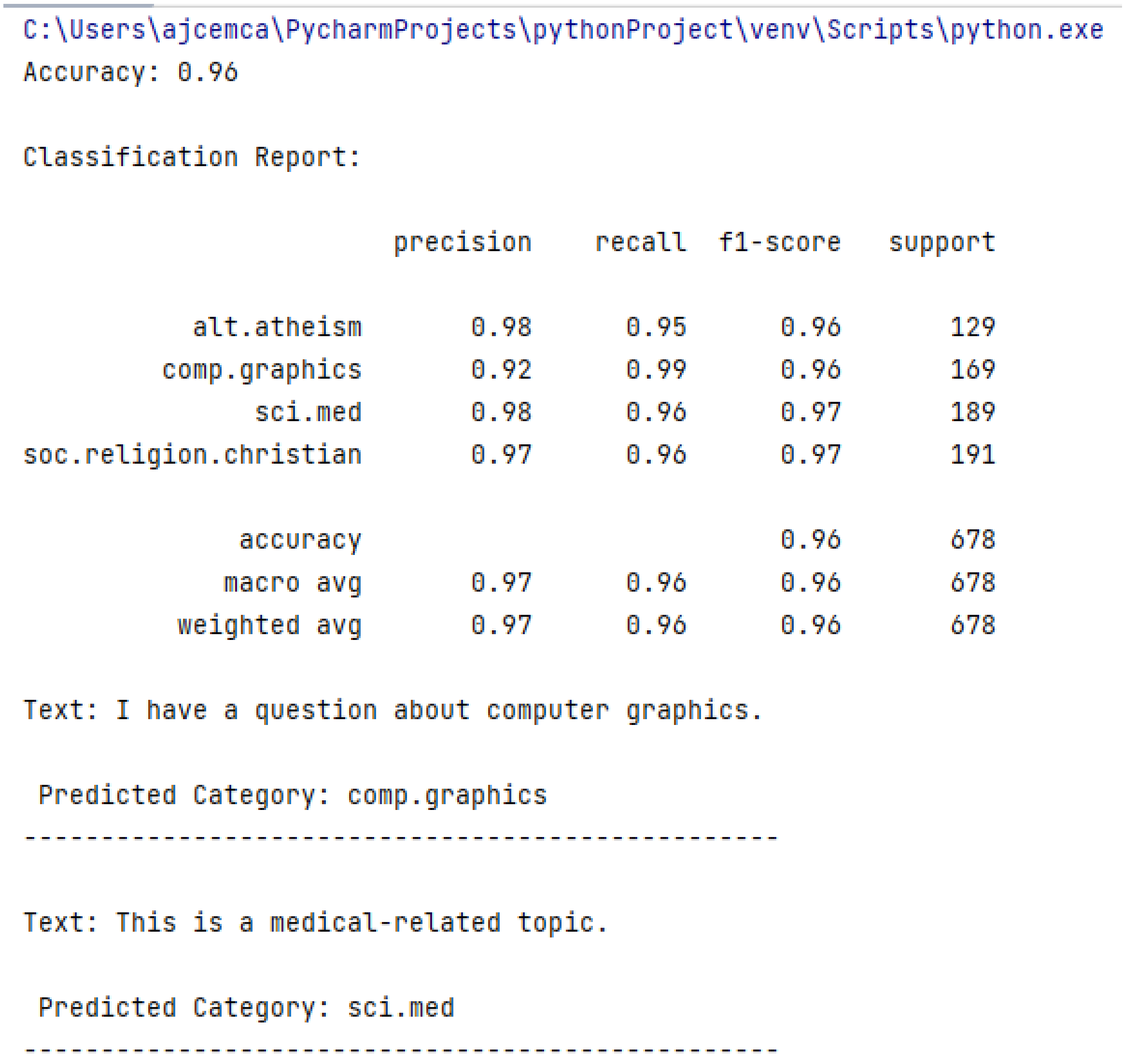
CO3

Use different packages and frameworks to implement test 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 classification_report, accuracy_score
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_size=0.3,
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)
class_report = classification_report(y_test, predictions, target_names=twenty_train.target_names)
print(f"Accuracy: {accuracy:.2f} \n")
print(f"Classification Report: \n")
print(class_report)
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):
    print(f"Text: {text}")
    predicted_category = twenty_train.target_names[new_predictions[i]]
    print(f"\n Predicted Category: {predicted_category}")
    print("------------------------------------------------- \n")
```

Output Screenshot



```
C:\Users\ajcemca\PycharmProjects\pythonProject\venv\Scripts\python.exe
Accuracy: 0.96

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

Text: I have a question about computer graphics.

 Predicted Category: comp.graphics
--------------------------------------------------

Text: This is a medical-related topic.

 Predicted Category: sci.med
--------------------------------------------------
```

Result

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

Experiment No:16

Aim

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

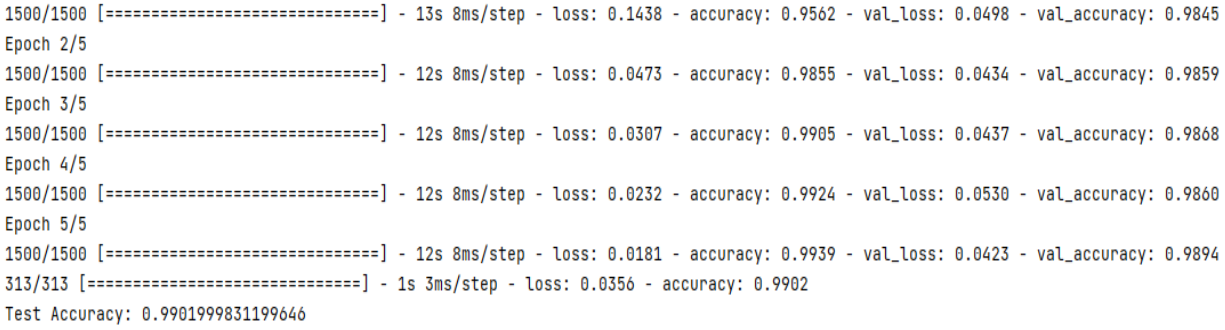
CO4

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,Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
(X_train,y_train),(X_test,y_test) = mnist.load_data()
X_train= X_train.reshape(-1,28,28,1)/255.0
X_test=X_test.reshape(-1,28,28,1)/ 255.0
y_train=to_categorical(y_train)
y_test= to_categorical(y_test)
model=Sequential([
   Conv2D(32,(3,3),activation='relu', input_shape=(28,28,1)),
   MaxPooling2D((2,2)),
   Conv2D(64,(3,3),activation='relu'),MaxPooling2D((2,2)),
   Flatten(),
   Dense(128,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(f'Test Accuracy: {accuracy}')
```

Output Screenshot:



```
1500/1500 [==============================] - 13s 8ms/step - loss: 0.1438 - accuracy: 0.9562 - val_loss: 0.0498 - val_accuracy: 0.9845
Epoch 2/5
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0473 - accuracy: 0.9855 - val_loss: 0.0434 - val_accuracy: 0.9859
Epoch 3/5
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0307 - accuracy: 0.9905 - val_loss: 0.0437 - val_accuracy: 0.9868
Epoch 4/5
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0232 - accuracy: 0.9924 - val_loss: 0.0530 - val_accuracy: 0.9860
Epoch 5/5
1500/1500 [==============================] - 12s 8ms/step - loss: 0.0181 - accuracy: 0.9939 - val_loss: 0.0423 - val_accuracy: 0.9894
313/313 [==============================] - 1s 3ms/step - loss: 0.0356 - accuracy: 0.9902
Test Accuracy: 0.9901999831199646
```

Result

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

Experiment No:17

Aim

Program to implement a simple web crawler using requests library

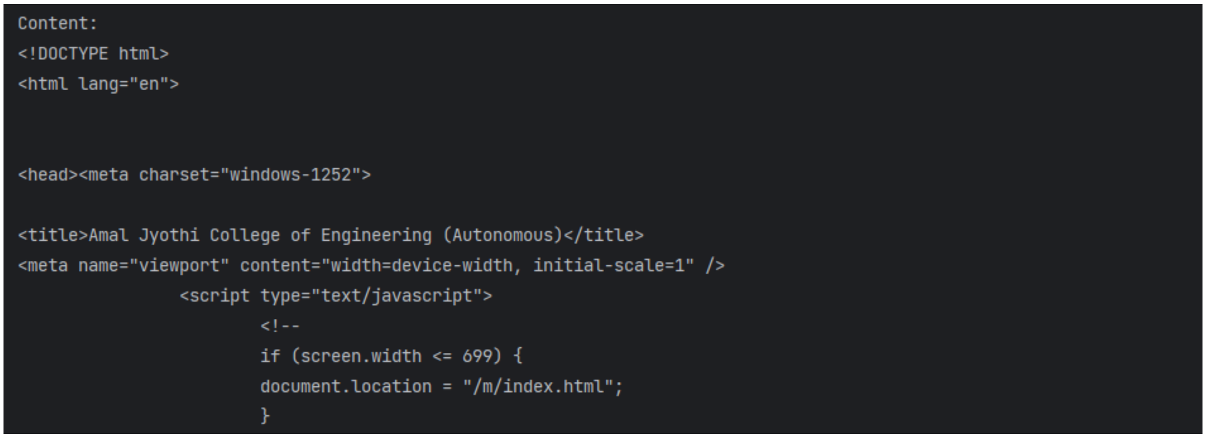
CO5

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

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";
                        }
```

Result

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

Experiment No:18

Aim

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

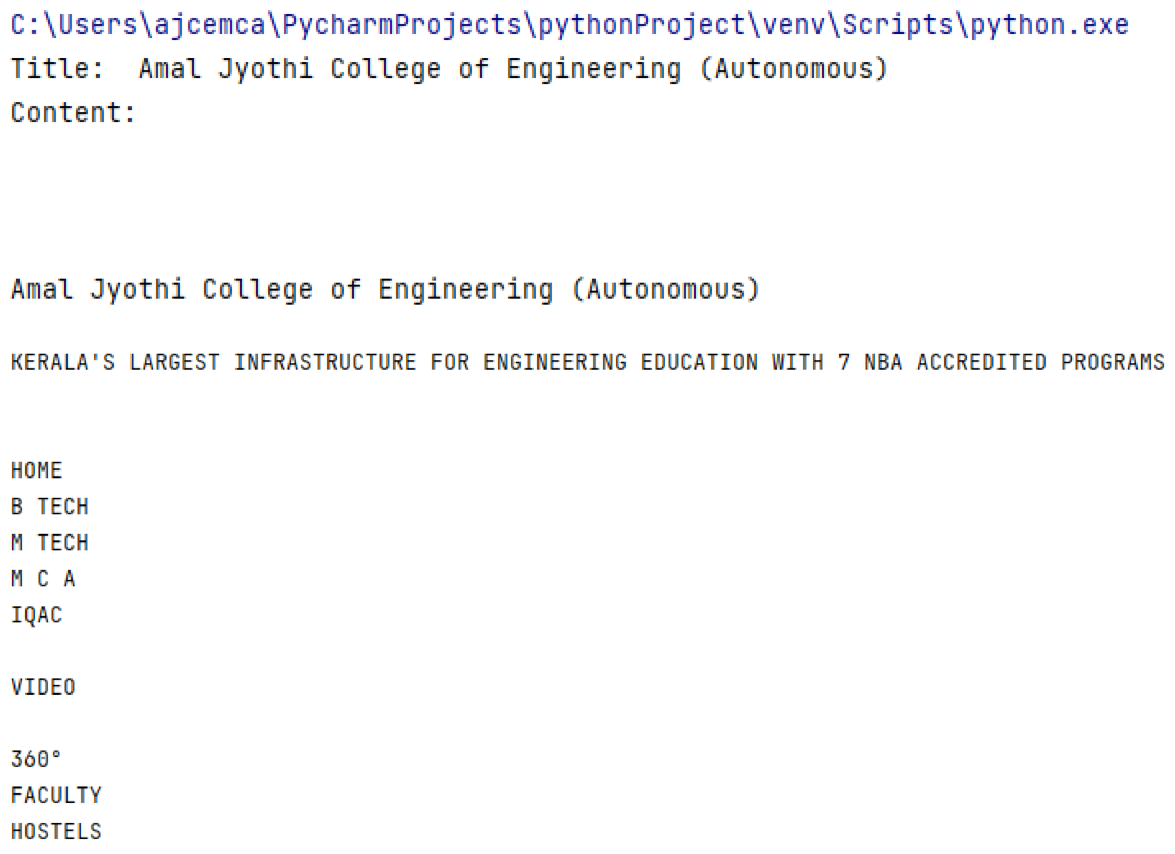
CO5

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

Procedure

```
import requests
from bs4 import BeautifulSoup
def simple_scraper_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_scrape = "https://ajce.in"
simple_scraper_with_bs(url_to_scrape)
```

Output Screenshot:



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




Amal Jyothi College of Engineering (Autonomous)

KERALA'S LARGEST INFRASTRUCTURE FOR ENGINEERING EDUCATION WITH 7 NBA ACCREDITED PROGRAMS


HOME
B TECH
M TECH
M C A
IQAC

VIDEO

360°
FACULTY
HOSTELS
```

Result

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

Experiment No:19

Aim

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

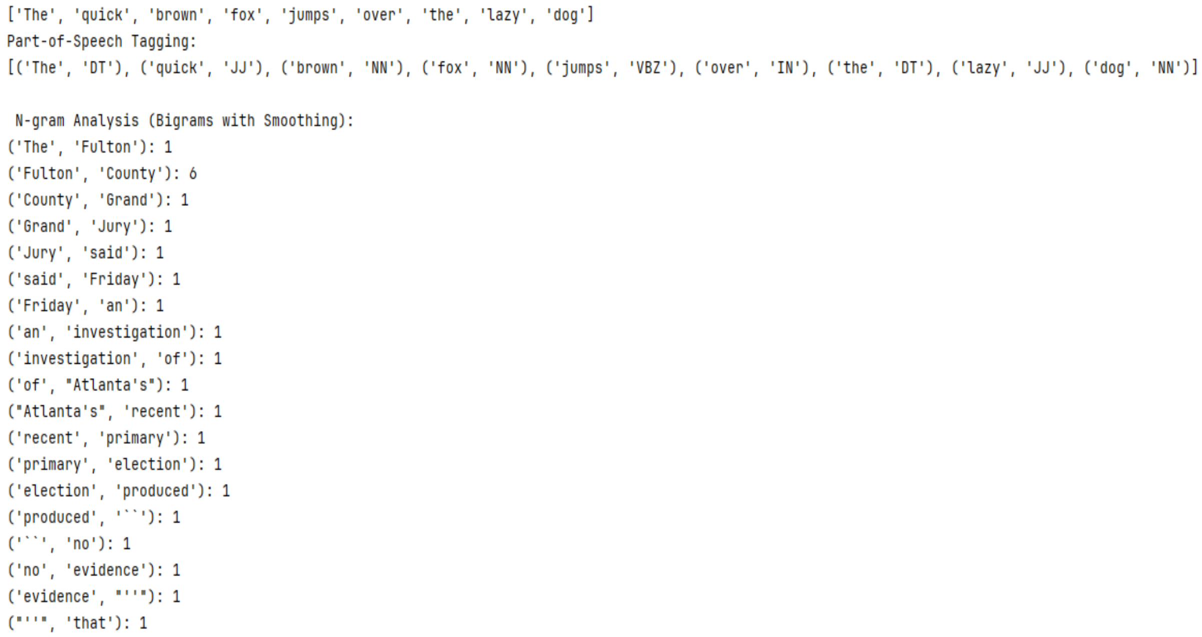
CO5

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
("Atlanta's", 'recent'): 1
('recent', 'primary'): 1
('primary', 'election'): 1
('election', 'produced'): 1
('produced', '``'): 1
('``', 'no'): 1
('no', 'evidence'): 1
('evidence', "''"): 1
("''", 'that'): 1
```

Result

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