20MCA241– DATA SCIENCE LAB

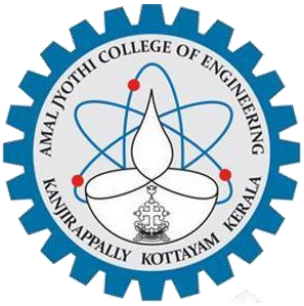
Lab Report Submitted By

VISHNU SADASIVAN

Reg. No.: AJC21MCA-2112

In Partial fulfillment for the Award of the Degree Of

MASTER OF COMPUTER APPLICATIONS (2 Year)
(MCA)
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY

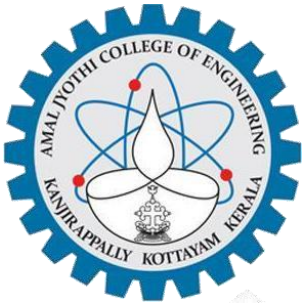
[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2022-2023

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 **VISHNU SADASIVAN(AJC21MCA-2112)** 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 2022-23.

Ms.Sruthimol Kurian
Lab In-Charge

Rev. Fr. Dr. Rubin Thottupuram Jose
Head of the Department

Internal Examiner

External Exam

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
CO2	Use different packages and frameworks to implement regression and classification algorithms.	60
CO3	Use different packages and frameworks to implement text classification using SVM and clustering using k-means	60
CO4	Implement convolutional neural network algorithm using Keras framework.	60
CO5	Implement programs for web data mining and natural language processing using NLTK	60

COURSE END SURVEY

CO	Survey Question	Answer Format
CO1	To what extend you are able to use different python packages to perform numerical calculations, statistical computations and data visualization?	Excellent/Very Good/Good Satisfactory/Needs improvement
CO2	To what extend you are able to use different packages and frameworks to implement regression and classification algorithms?	Excellent/Very Good/Good Satisfactory/Needs improvement

CO3	To what extend you are able to use different packages and frameworks to implement text classification using SVM and clustering using K-means?	Excellent/Very Good/Good Satisfactory/Needs improvement
CO4	To what extend you are able to implement convolutional neural network algorithm using Keras framework?	Excellent/Very Good/Good Satisfactory/Needs improvement
CO5	To what extend you are able to implement programs for web data mining and natural language processing using NLTK?	Excellent/Very Good/Good Satisfactory/Needs improvement

CONTENT

Program No: 01

Aim:

Create a student table with columns Roll.no, Name, age, marks using pandas and do the following

a) select the top 2 rows
b) filter data based on some condition with mark > 80
c) filter in names first name start with 'v' then remaining.

CO1

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

Program & Output:

```
import pandas as pd
mark= pd.DataFrame({'rollno':[72,45,11,44,32],'name': ["manu","vishnu","don","sathya","ram"],
'age':[18,33,17,22,32],'marks':[66,86,67,81,79]})
print(mark.head(2))
```

```
   rollno    name  age  marks
0      72    manu   18     66
1      45  vishnu   33     86
```

```
mark [mark['marks'] > 80]
```

	rollno	name	age	marks
1	45	vishnu	33	86
3	44	sathya	22	81

```
data[data['name'].str.startswith('N')]
```

	rollno	name	age	marks
1	45	vishnu	33	86

Result:

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

Program No: 02

Aim:

Numpy array creation and basic operations, Initialization, array indexing.

CO1

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

Program & Output:

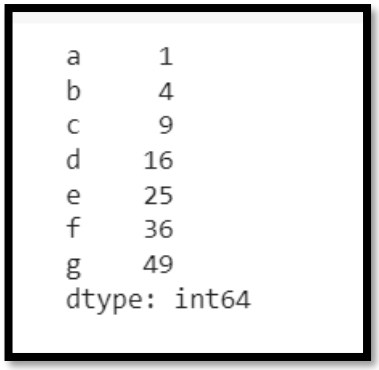
```
import pandas as pd
import numpy as np
arr=np.array(['a','b','c','d'])
s=pd.Series(arr,
index=['first','second','third','fourth'])
print(s)
```

```
first      a
second     b
third      c
fourth     d
dtype: object
```

```
import pandas as pd
p=pd.Series([1,2,3,4,5])
print(p)
print("multlipling all values in series by 2")
print(p*2)
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
multlipling all values in series by 2
0     2
1     4
2     6
3     8
4    10
```

```
print(pd.Series(np.array([1,2,3,4,5,6,7]), index=['a','b','c','d','e','f','g'])**2)
```



```
a     1
b     4
c     9
d    16
e    25
f    36
g    49
dtype: int64
```

Result

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

Program No: 03

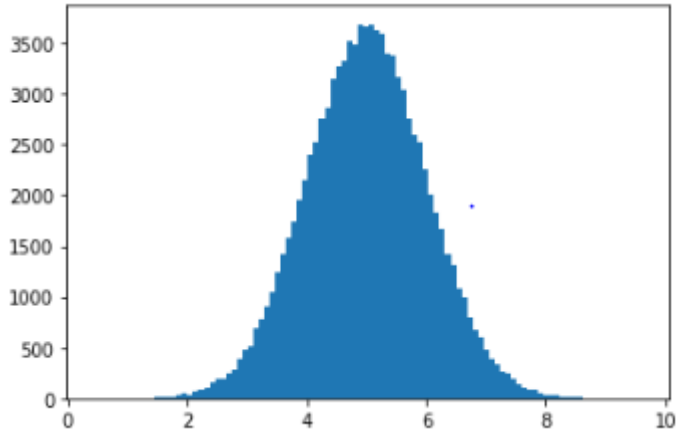
Aim: Plot a graph by matplotlib library

CO1

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

Program & Output:

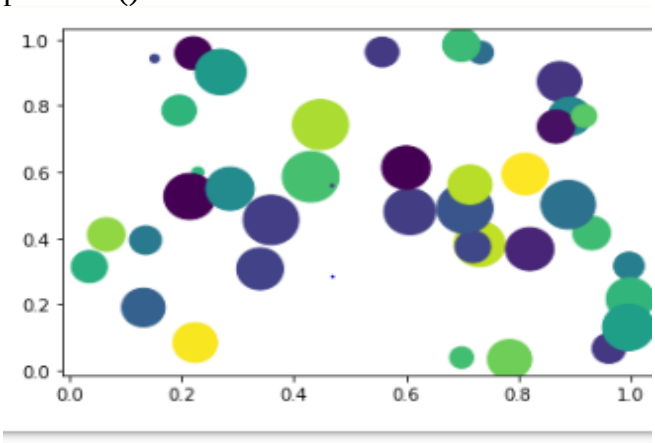
```
import numpy
import matplotlib.pyplot as plt

x = numpy.random.normal(5.0, 1.0, 100000)

plt.hist(x, 100)
plt.show()
```



Bubble Chart

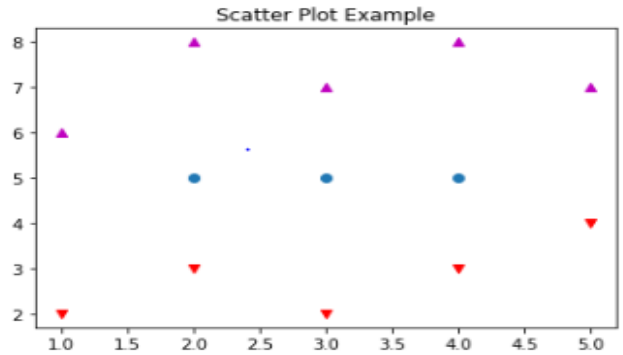
```
import matplotlib.pyplot as plt
import numpy as np

# create data
x = np.random.rand(40)
y = np.random.rand(40)
z = np.random.rand(40)
colors = np.random.rand(40)
# use the scatter function
```

```
plt.scatter(x, y, s=z*1000,c=colors)
plt.show()
```



Scatter Plot

```
import matplotlib.pyplot as plt

x1 = [2, 3, 4]
y1 = [5, 5, 5]

x2 = [1, 2, 3, 4, 5]
y2 = [2, 3, 2, 3, 4]
y3 = [6, 8, 7, 8, 7]

plt.scatter(x1, y1)
plt.scatter(x2, y2, marker='v', color='r')
plt.scatter(x2, y3, marker='^', color='m')
plt.title('Scatter Plot Example')
plt.show()
```



Result

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

Program No: 04

Aim:

Perform all matrix operation using python (using numpy)

CO1:

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

Program & Output:

```
import numpy as np
a = np.array([1, 2, 3])   # Creating the rank 1 array
print("type: " ,type(a))           # Prints "<class 'numpy.ndarray'>"
print("shape: " ,a.shape)           # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Changing an element of the array
print(a)                  # Printing the "[5, 2, 3]"
b = np.array([[1,2,3],[4,5,6]])    # Creating the rank 2 array
print("\n shape of b:",b.shape)                    # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])   # Printing the  "1 2 4"
a = np.zeros((3,3))   # Create the array of all zeros
print("All zeros matrix:\n  " ,a)
b = np.ones((1,2))    # Creating the array of all ones
print("\nAll ones matrix:\n  " ,b)            # Prints "[[ 1.  1.]]"
d = np.eye(2)        # Creating the 2x2 identity matrix
print("\n identity matrix: \n",d)
e = np.random.random((2,2))  # Creating the array filled with random values
print("\n random matrix: \n",e)
```

```
shape:  (3,)
1 2 3
[5 2 3]

 shape of b: (2, 3)
1 2 4
All zeros matrix:
   [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]

All ones matrix:
   [[1. 1.]]

 identity matrix:
 [[1. 0.]
 [0. 1.]]

 random matrix:
 [[0.03208804 0.89066577]
 [0.71424012 0.51353052]]
```

Result:

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

Program No: 05

Aim: Program to Perform SVD (Singular Value Decomposition) in Python.

CO1

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

Program & Output:

```
# Singular-value decomposition

from numpy import array
from scipy.linalg import svd
# define a matrix
A = array([[1, 2], [3, 4], [5, 6]])
print("A: \n%s" %A)
# SVD
U, s, VT = svd(A)
print("\nU: \n%s" %U)
print("\ns: \n %s" %s)
print("\nV^T: \n %s" %VT)
```

Output:

```
A:

[[1 2]
 [3 4]
 [5 6]]

U:
[[-0.2298477   0.88346102  0.40824829]
 [-0.52474482  0.24078249 -0.81649658]
 [-0.81964194 -0.40189603  0.40824829]]
s:
 [9.52551809 0.51430058]

V^T:
 [[-0.61962948 -0.78489445]
 [-0.78489445  0.61962948]]
```

Result

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

Program No: 06

Aim:

Program to implement k-NN classification 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.

Program & Output:

```
# KNN Algorithm using IRIS Dataset

import random
import csv
split = 0.66

with open('iris_dataset.txt') as csvfile:
    lines = csv.reader(csvfile)
    dataset = list(lines)

random.shuffle(dataset)

div = int(split * len(dataset))
train = dataset [:div]
test = dataset [div:]

import math
# square root of the sum of the squared differences between the two arrays of numbers
def euclideanDistance(instance1, instance2, length):
  distance = 0
  for x in range(length):
    distance += pow((float(instance1[x]) - float(instance2[x])), 2)
  return math.sqrt(distance)

import operator
#distances = []
def getNeighbors(trainingSet, testInstance, k):
  distances = []
  length = len(testInstance)-1
  for x in range(len(trainingSet)):
    dist = euclideanDistance(testInstance, trainingSet[x], length)
```

```
    distances.append((trainingSet[x], dist))
  distances.sort(key=operator.itemgetter(1))
  neighbors = []
  for x in range(k):
    neighbors.append(distances[x][0])
  return neighbors

classVotes = {}
def getResponse(neighbors):
  for x in range(len(neighbors)):
    response = neighbors[x][-1]
    if response in classVotes:
      classVotes[response] += 1
    else:
      classVotes[response] = 1
  sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1), reverse=True)
  return sortedVotes[0][0]

def getAccuracy(testSet, predictions):
  correct = 0
  for x in range(len(testSet)):
    if testSet[x][-1] == predictions[x]:
      correct += 1
  return (correct/float(len(testSet))) * 100.0

predictions=[]
k = 3

for x in range(len(test)):
    neighbors = getNeighbors(train, test[x], k)
    result = getResponse(neighbors)
    predictions.append(result)
    print('> predicted=' + repr(result) + ', actual=' + repr(test[x][-1]))

accuracy = getAccuracy(test, predictions)
print('Accuracy: ' + repr(accuracy) + '%')
```

Output:

```
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-virginica', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-virginica', actual='Iris-virginica'
```

> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-setosa'
> predicted='Iris-setosa', actual='Iris-setosa'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-setosa', actual='Iris-setosa'
> predicted='Iris-setosa', actual='Iris-setosa'
> predicted='Iris-setosa', actual='Iris-versicolor'
> predicted='Iris-setosa', actual='Iris-setosa'
> predicted='Iris-setosa', actual='Iris-virginica'
> predicted='Iris-setosa', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-versicolor'
> predicted='Iris-versicolor', actual='Iris-virginica'
> predicted='Iris-versicolor', actual='Iris-virginica'

Accuracy: 41.17647058823529%

Result:

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

Program No: 07

Aim:

Program to implement Naive 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.

Program & Output:

```
# Naive Bayes using Iris Dataset

from sklearn.datasets import load_iris
iris = load_iris()
# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target
# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
# making predictions on the testing set
y_pred = gnb.predict(X_test)
# comparing actual response values (y_test) with predicted response values (y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*1
00)
```

```
Gaussian Naive Bayes model accuracy(in %): 95.0
```

Result:

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

Program No: 08

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.

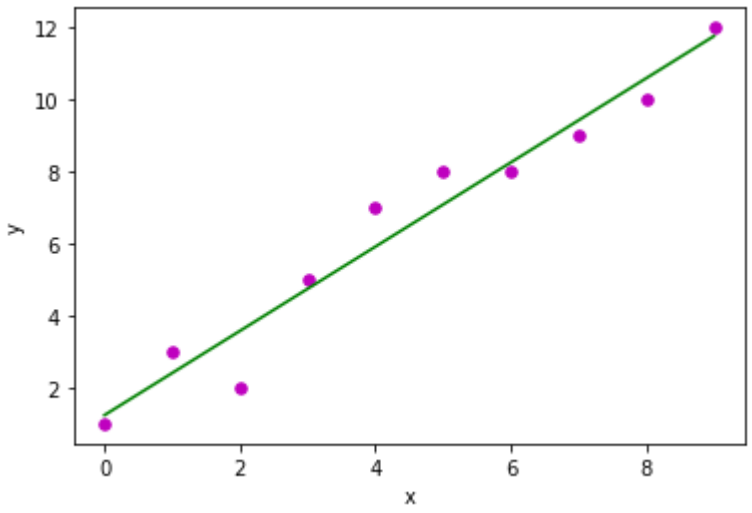
Program & Output:

```
import numpy as np
import matplotlib.pyplot as plt
def estimate_coef(x, y):
    # number of observations/points
    n = np.size(x)

    # mean of x and y vector
    m_x = np.mean(x)
    m_y = np.mean(y)

    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x

    return (b_0, b_1)

def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m",
               marker = "o", s = 30)

    # predicted response vector
    y_pred = b[0] + b[1]*x

    # plotting the regression line
    plt.plot(x, y_pred, color = "g")

    # putting labels
```

```
    plt.xlabel('x')
    plt.ylabel('y')

    # function to show plot
    plt.show()

def main():
    # observations / data
    x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

    # estimating coefficients
    b = estimate_coef(x, y)
    print(b)
    print("Estimated coefficients:\nb_0 = {}  \
        \nb_1 = {}".format(b[0], b[1]))

    # plotting regression line
    plot_regression_line(x, y, b)

if __name__ == "__main__":
    main()
```

OUTPUT:

(1.2363636363636363, 1.1696969696969697)
Estimated coefficients:
b_0 = 1.2363636363636363
b_1 = 1.1696969696969697



```
# Multiple Linear Regression using custom data

import numpy as np
from sklearn.linear_model import LinearRegression

x = [[0, 1], [5, 1], [15, 2], [25, 5], [35, 11], [45, 15], [55, 34], [60, 35]]
y = [4, 5, 20, 14, 32, 22, 38, 43]
x, y = np.array(x), np.array(y)
print(x)
print(y)
model = LinearRegression().fit(x, y)
r_sq = model.score(x, y)
print(f"coefficient of determination: {r_sq}")
print(f"intercept: {model.intercept_}")
print(f"coefficients: {model.coef_}")
y_pred = model.predict(x)
print(f"predicted response:\n{y_pred}")
x_new = np.arange(10).reshape((-1, 2))
print(x_new)
y_new = model.predict(x_new)
y_new
```

```
[[ 0  1]
 [ 5  1]
 [15  2]
 [25  5]
 [35 11]
 [45 15]
 [55 34]
 [60 35]]
[ 4  5 20 14 32 22 38 43]
coefficient of determination: 0.8615939258756775
intercept: 5.52257927519819
coefficients: [0.44706965 0.25502548]
predicted response:
[ 5.77760476  8.012953   12.73867497 17.9744479  23.97529728 29.4660957
 38.78227633 41.27265006]
[[0 1]
 [2 3]
 [4 5]
 [6 7]
 [8 9]]
array([ 5.77760476,  7.18179502,  8.58598528,  9.99017554, 11.3943658 ])
```

Result:

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

Program No: 09

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 regression and classification algorithms.

Program & Output:

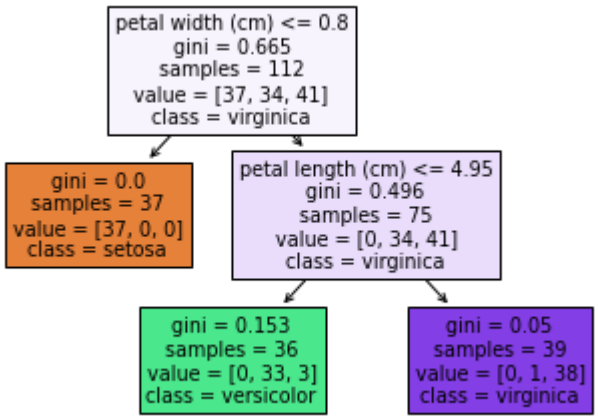
```
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn.datasets import load_iris
data = load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

X_train, X_test, Y_train, Y_test = train_test_split(df[data.feature_names], df['target'],
random_state=0)

# Step 1: Import the model you want to use
# This was already imported earlier in the notebook so commenting out
# from sklearn.tree import DecisionTreeClassifier
# Step 2: Make an instance of the Model
clf = DecisionTreeClassifier(max_depth=2,
                             random_state=0)
# Step 3: Train the model on the data
clf.fit(X_train, Y_train)
# Step 4: Predict labels of unseen (test) data
# Not doing this step in the tutorial
clf.predict(X_test)
# tree.plot_tree(clf);
fn = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
cn = ['setosa', 'versicolor', 'virginica']
# fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(4, 4), dpi=300)
tree.plot_tree(clf,
               feature_names=fn,
```

```
            class_names=cn,
            filled=True
            )
y_pred = clf.predict(X_test)
print("Train data accuracy:",accuracy_score(y_true = Y_train, y_pred=clf.predict(X_train)))
print("Test data accuracy:",accuracy_score(y_true = Y_test, y_pred=y_pred))
plt.show()
```

OUTPUT:

Train data accuracy: 0.9642857142857143

Test data accuracy: 0.8947368421052632



Result:

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

Program No: 10

Aim:

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

CO3

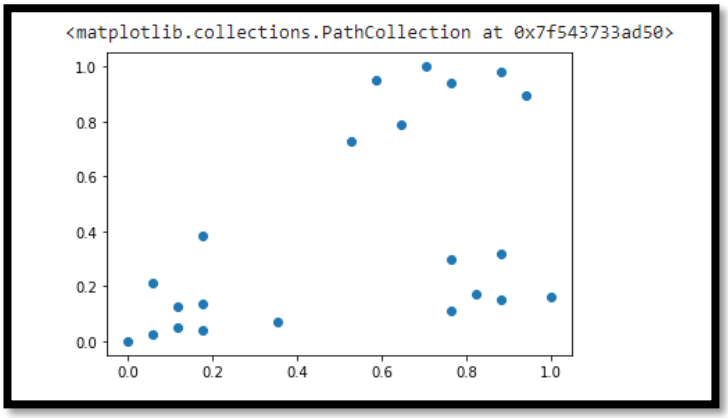
Use different packages and frameworks to implement text classification using SVM and clustering using k-means

Program & Output:

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
df = pd.read_csv('income.csv')
df.head()
```

	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000

```
scaler = MinMaxScaler()
scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])
scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])
plt.scatter(df.Age, df['Income($)'])
```




```
km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age', 'Income($)']])
y_predicted
```

```
array([1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0],
      dtype=int32)
```

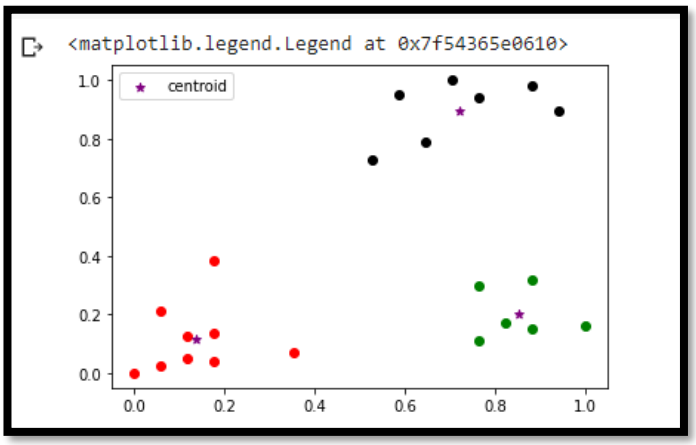
```
df['cluster'] = y_predicted
df.head()
```

	Name	Age	Income($)	cluster
0	Rob	0.058824	0.213675	1
1	Michael	0.176471	0.384615	1
2	Mohan	0.176471	0.136752	1
3	Ismail	0.117647	0.128205	1
4	Kory	0.941176	0.897436	2

```
km.cluster_centers_
```

```
array([[0.85294118, 0.2022792 ],
       [0.1372549 , 0.11633428],
       [0.72268908, 0.8974359 ]])
```

```
df1 = df[df.cluster==0]
df2 = df[df.cluster==1]
df3 = df[df.cluster==2]
plt.scatter(df1.Age, df1['Income($)'], color = 'green')
plt.scatter(df2.Age, df2['Income($)'], color = 'red')
plt.scatter(df3.Age, df3['Income($)'], color = 'black')
plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1], color='purple', marker = '*', label
= 'centroid')
plt.legend()
```



Result:

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

Program No: 11

Aim:

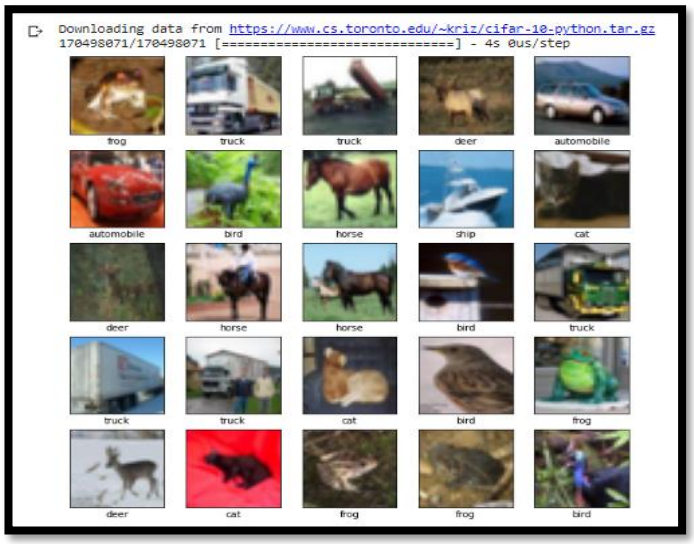
Implementation of CNN using keras network

CO4

Implement convolutional neural network algorithm using Keras framework.

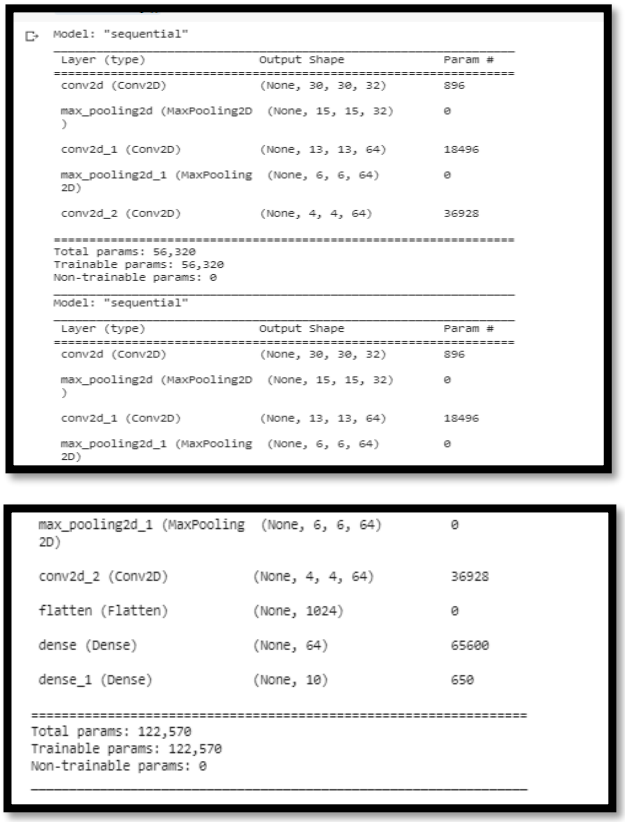
Program & Output:

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```




```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
```

```
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.summary()
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
model.summary()
```

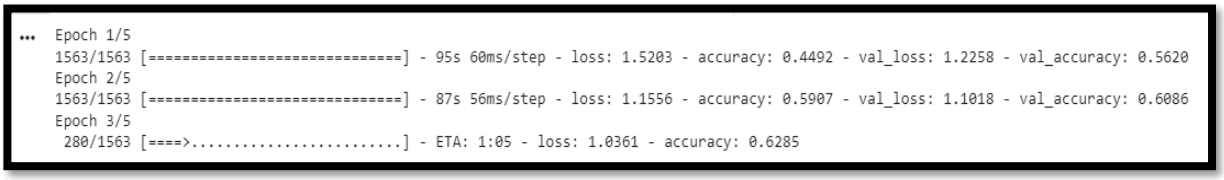


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

=================================================================
Total params: 56,320
Trainable params: 56,320
Non-trainable params: 0
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)
```



```
 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=5,
                    validation_data=(test_images, test_labels))
```



```
Epoch 1/5
1563/1563 [==============================] - 95s 60ms/step - loss: 1.5203 - accuracy: 0.4492 - val_loss: 1.2258 - val_accuracy: 0.5620
Epoch 2/5
1563/1563 [==============================] - 87s 56ms/step - loss: 1.1556 - accuracy: 0.5907 - val_loss: 1.1018 - val_accuracy: 0.6086
Epoch 3/5
 280/1563 [====>.........................] - ETA: 1:05 - loss: 1.0361 - accuracy: 0.6285
```

Result:

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

Program No: 12

Aim:

Program to implement scrap of any website

CO5

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

Program & Output:

```
import requests
from bs4 import BeautifulSoup
URL = "http://www.ajce.in"
r = requests.get(URL)
soup = BeautifulSoup(r.content, 'html5lib')
print(soup.prettify())
```

Output:

```
<!DOCTYPE html>
<html lang="en-us" prefix="og: http://ogp.me/ns#">
 <head>
  <meta charset="utf-8"/>
  <meta content="Data Structures,Algorithms,Python,Java,C,C++,JavaScript,Android Development,SQL,Data Science,Machine Learning,PHP,Web Development,System Design,Tutorial,Technical Blogs,Interview Experience,Interview Preparation,Programming,Competitive Programming,SDE Sheet,Job-a-thon,Coding Contests,GATE CSE,HTML,CSS,React,NodeJS,Placement,Aptitude,Quiz,Computer Science,Programming Examples,GeeksforGeeks Courses,Puzzles" name="keywords"/>
  <meta content="width=device-width,initial-scale=1,maximum-scale=1" name="viewport"/>
  <meta content="A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive programming/company interview Questions." name="description" property="og:description"/>
  <meta content="https://www.geeksforgeeks.org/" property="og:url"/>
  <link href="https://media.geeksforgeeks.org/wp-content/cdn-uploads/gfg_favicon.png" rel="shortcut icon" type="image/x-icon"/>
  <meta content="#308D46" name="theme-color"/>
  <meta content="https://media.geeksforgeeks.org/wp-content/cdn-uploads/gfg_200x200-min.png" name="image" property="og:image"/>
  <meta content="image/png" property="og:image:type"/>
  <meta content="200" property="og:image:width"/>
  <meta content="200" property="og:image:height"/>
```

<meta content="xo7t4ve2wn3ywfkjdvwbrk01pvdond" name="facebook-domain-verification"/>
<script defer="" src="https://apis.google.com/js/platform.js">
</script>
<script async="" src="//cdnjs.cloudflare.com/ajax/libs/require.js/2.1.14/require.min.js">
</script>
<title>
GeeksforGeeks | A computer science portal for geeks
</title>
<link href="http://gmpg.org/xfn/11" rel="profile"/>
<link href="" rel="pingback"/>
<script type="application/ld+json">

{"@context":"http://schema.org","@type":"Organization","name":"GeeksforGeeks","url":"https://www.geeksforgeeks.org/","logo":"https://media.geeksforgeeks.org/wp-content/cdn-uploads/20200817185016/gfg_complete_logo_2x-min.png","description":"A computer science portal for geeks. It contains well written, well thought and well explained computer science and programming articles, quizzes and practice/competitive programming/company interview Questions.","founder":[{"@type":"Person","name":"Sandeep Jain","url":"https://in.linkedin.com/in/sandeep-jain-b3940815"}],"sameAs":["https://www.facebook.com/geeksforgeeks.org/","https://twitter.com/geeksforgeeks","https://www.linkedin.com/company/1299009","https://www.youtube.com/geeksforgeeksvideos/"]}
</script>
<script>
var arrPostCat=new Array,arrPostCatName="GBlog",tIds,domain,arrPost,post_id,post_type,post_slug,ip,post_title,post_status,practiceAPIURL,practiceURL,post_date,commentSysUrl,link_on_code_run,link_search_modal_top,isAdminLoggedIn,postAdApiUrlString;arrPostCat.push('1710'),arrPostCat.push('6256'),tIds="1710,6256",domain=1,arrPost=new Array,post_id="0",post_type="post",post_slug=window.location.href,ip="35.86.2.185",post_title="What is System Design – Learn System Design",post_status="publish",practiceAPIURL="https://practiceapi.geeksforgeeks.org/",practiceURL="https://practice.geeksforgeeks.org/",post_date="2022-11-27 10:08:42",commentSysUrl="https://discuss.geeksforgeeks.org/commentEmbedV2.js",link_on_code_run='

Improve your Coding Skills with Practice',link_search_modal_top='',isAdminLoggedIn=0,postAdApiUrlString='1710/6256/'
</script>
<meta content="en_US" property="og:locale"/>
<meta content="website" property="og:type"/>
<meta content="GeeksforGeeks | A computer science portal for geeks" property="og:title"/>
<meta content="GeeksforGeeks" property="og:site_name"/>
<script type="application/ld+json">
<a aria-label="Logo" class="header-main__logo" href="https://www.geeksforgeeks.org/">

```
    <img alt="geeksforgeeks" class="gfg_logo_img" src="https://media.geeksforgeeks.org/gfg-gg-logo.svg"/>
   </div>
  </a>
  <div class="header-main__container">
   <span class="hamburger-menu">
    <span class="gfg-burger-1">
    </span>
    <span class="gfg-burger-2">
    </span>
    <span class="gfg-burger-3">
    </span>
   </span>
   <ul class="header-main__list">
    <li aria-expanded="true" class="header-main__list-item" data-expandable="true" data-parent="false">
     <span>
      Courses
     </span>
     <i class="gfg-icon gfg-icon_arrow-down gfg-icon_header">
     </i>
     <ul class="mega-dropdown">
      <li aria-expanded="true" class="mega-dropdown__list-item" data-expandable="true" data-parent="true">
       <span>
        For Working Professionals
       </span>
       <i class="gfg-icon gfg-icon_arrow-right">
       </i>
       <ul class="mega-dropdown">
        <li aria-expanded="false" class="mega-dropdown__list-item" data-child="true" data-expandable="false">
         <a href="https://practice.geeksforgeeks.org/courses/geeks-classes-live?utm_source=gfg&amp;utm_medium=Submenu&amp;utm_campaign=courses-submenu" target="_self">
          Data Structure &amp;amp; Algorithm Classes (Live)
         </a>
        </li>
        <li aria-expanded="false" class="mega-dropdown__list-item" data-child="true" data-expandable="false">

</html>
```

Result:

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

Program No: 13

Aim:

Program for Natural Language Processing which performs n-grams(Using inbuilt functions)

CO5

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

Program & Output:

```
import nltk
from nltk.util import ngrams
text = "Hello ram! Long time no see";
Ngrams = ngrams(sequence = nltk.wordpunct_tokenize(text), n=2)
for grams in Ngrams:
  print(grams)
```

```
('Hello', 'ram')
('ram', '!')
('!', 'Long')
('Long', 'time')
('time', 'no')
('no', 'see')
```

Result:

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

Program No: 14

Aim:

Program for Natural Language Processing which perform parts of speech tagging.

CO5

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

Program & Output:

```
import nltk
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
exptagger.tag_sents([['Hi', ','], ['where', 'are', 'you', 'from','?']])
```

```
[[('Hi', 'NN'), (',', 'NN')],
 [('where', 'NN'), ('are', 'NN'), ('you', 'NN'), ('from', 'NN'), ('?', 'NN')]]
```

```
import nltk
from nltk.tag import untag
untag([('Tutorials', 'NN'), ('Point', 'NN')])
```

```
['Tutorials', 'Point']
```

```
import nltk
# import all the resources for Natural Language Processing with Python
nltk.download("book")
```

```
[nltk_data] Downloading collection 'book'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
```

```
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Unzipping help/tagsets.zip.
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection book
True
```

#Take a sentence and tokenize into words. Then apply a part-of-speech tagger.
sentence = """At eight o'clock on Thursday morning
Arthur didn't feel very good."""
tokens = nltk.word_tokenize(sentence)
print(tokens)
tagged = nltk.pos_tag(tokens)
print(tagged)

```
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning', 'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'NN'), ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'), ('Arthur', 'NNP'), ('did', 'VB
```

text ="learn php from guru99 and make study easy".split()
print("After Split:",text)
tokens_tag = nltk.pos_tag(text)
print("After Token:",tokens_tag)

```
After Split: ['learn', 'php', 'from', 'guru99', 'and', 'make', 'study', 'easy']
After Token: [('learn', 'JJ'), ('php', 'NN'), ('from', 'IN'), ('guru99', 'NN'), ('and', '
```

Result:

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

Program No: 15

Aim:

Data pre-processing with NLTK

a) Counting Tags
b) Bigrams
c) Trigrams
d) Stop Words
e) Stemming

CO5

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

Program & Output:

```
!pip install -q wordcloud
import wordcloud
import nltk
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')
import pandas as pd
import unicodedata
import numpy as np
import string
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
import nltk
text = "A Computer Science portal for geeks. It contains well written, well thought and well expl
ained computer science"
lower_case = text.lower()
tokens = nltk.word_tokenize(lower_case)
tags = nltk.pos_tag(tokens)
```

```
counts = nltk.Counter( tag for word,  tag in tags)
print(counts)
```

```
Counter({'NN': 6, 'RB': 3, 'VBN': 3, 'DT': 1, 'IN': 1, '.': 1, 'PRP': 1, 'VBZ': 1, ',': 1, 'CC': 1})
```

```
import nltk
text = "GeeksforGeeks is a one-stop destination for programmers"
Tokens = nltk.word_tokenize(text)
output = list(nltk.bigrams(Tokens))
print(output)
```

```
[('GeeksforGeeks', 'is'), ('is', 'a'), ('a', 'one-stop'), ('one-stop', 'destination'), ('destination', 'for'), ('for', 'programmers')]
```

```
import nltk
text = "GeeksforGeeks is a one-stop destination for programmers"
Tokens = nltk.word_tokenize(text)
output = list(nltk.trigrams(Tokens))
print(output)
```

```
[('GeeksforGeeks', 'is', 'a'), ('is', 'a', 'one-stop'), ('a', 'one-stop', 'destination'), ('one-stop', 'destination', 'for'), ('destination', 'for', 'programmers'
```

```
from nltk.corpus import stopwords
from nltk.corpus import stopwords
print(stopwords.words('english'))
en_stopwords = stopwords.words('english')
def remove_stopwords(text):
    result = []
    for token in text:
        if token not in en_stopwords:
            result.append(token)
    return result
text = "this is the only solution of that question".split()
remove_stopwords(text)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your',
['solution', 'question']
```

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
ps = PorterStemmer()
sentence = "GeeksforGeeks is a one-stop destination for programmers"
words = word_tokenize(sentence)
for w in words:
   print(w, " : ", ps.stem(w))
```

```
GeeksforGeeks  :  geeksforgeek
is  :  is
a  :  a
one-stop  :  one-stop
destination  :  destin
for  :  for
programmers  :  programm
```

Result:

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