20MCA241– DATA SCIENCE LAB

*Lab Report Submitted By*

**SANIO LUKE SEBASTIAN**

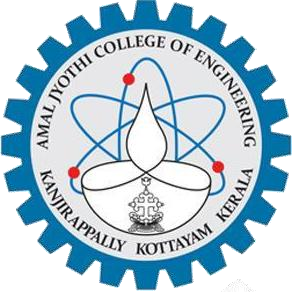
**Reg. No.: AJC21MCA-2093**

*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**

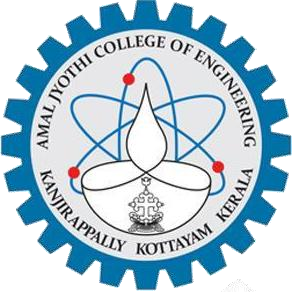
## [Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with ‘A’ grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

**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 **SANIO LUKE SEBASTIAN (AJC21MCA-2093)** 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 2021-22.

**Sr. Elsin Chakkalackal S.H. Rev. Fr. Dr. Rubin Thottupurathu Jose**

**Lab In-Charge 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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No** | **Content** | **CO** | **Date** | **Page No.** |
| **1** | Data Handling with Pandas | **CO1** | 12-09-2022 | 6 |
| **2** | Basic operation with Numpy | **CO1** | 22-08-2022 | 7-8 |
| **3** | Data visualization | **CO1** | 25-08-2022 | 9-10 |
| **4** | Matrix operation using Numpy | **CO1** | 11-08-2022 | 11 |
| **5** | Program to perform SVD | **CO1** | 11-08-2022 | 12 |
| **6** | Implementation of KNN- Classification | **CO2** | 22-09-2022 | 13-15 |
| **7** | Implementation of Naive-Bayes Classification | **CO2** | 26-09-2022 | 16 |
| **8** | Program to handle Multiple Linear Regression | **CO2** | 10-10-2022 | 17-19 |
| **9** | Implementation of Decision Tree Classification | **CO3** | 17-10-2022 | 20-21 |
| **10** | Implementation of k- means clustering | **CO3** | 20-10-2022 | 22-24 |
| **11** | Implementation of CNN using Keras Network | **CO4** | 27-10-2022 | 25-26 |
| **12** | Scraping of any website | **CO5** | 31-10-202 | 27-30 |
| **13** | Performs n- grams using NLP | **CO5** | 03-11-2022 | 31 |
| **14** | Perform parts of speech tagging using NLP | **CO5** | 07-11-2022 | 32-33 |
| **15** | Data pre-processing using NLTK | **CO5** | 14-11-2022 | 34-36 |

**Program No: 01**

**Aim:**

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

1. select the top 2 rows
2. filter data based on some condition with mark > 80
3. filter in names first name start with ‘N’ then remaining.

**CO1**

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

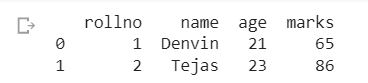
data visualization

**Program & Output:**

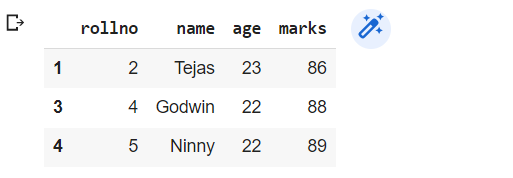
import pandas as pd

data= pd.DataFrame({'rollno':[1,2,3,4,5],'name': ["Denvin","Tejas","Avil","Godwin","Ninny"],'age':[21,23,22,22,22],'marks':[65,86,47,88,89]})

print(data.head(2))

****

data[data['marks'] > 80]

****

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

****

**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

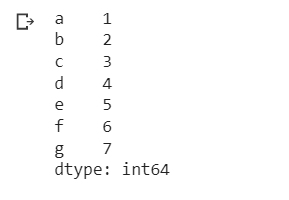
data visualization

**Program & Output:**

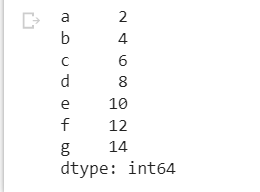
import pandas as pd

import numpy as np

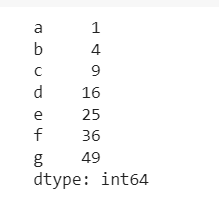
print(pd.Series(np.array([1,2,3,4,5,6,7]), index=['a','b','c','d','e','f','g']))



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



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



**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 pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

csv\_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

col\_names = ['Sepal\_Length','Sepal\_Width','Petal\_Length','Petal\_Width','Class']

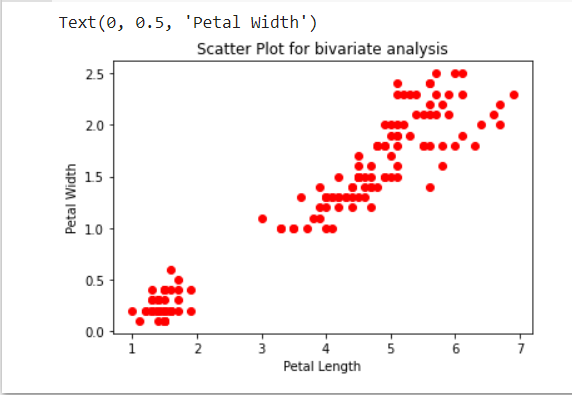
iris =  pd.read\_csv(csv\_url, names = col\_names)

plt.scatter(iris['Petal\_Length'],iris['Petal\_Width'],color='red')

plt.title("Scatter Plot for bivariate analysis")

plt.xlabel("Petal Length")

plt.ylabel("Petal Width")



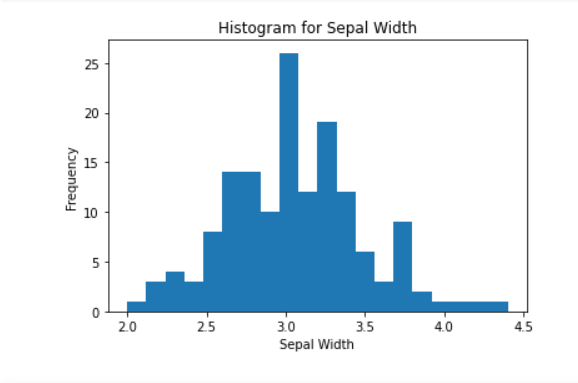
plt.hist(iris['Sepal\_Width'],bins=20)

plt.title("Histogram for Sepal Width")

plt.xlabel('Sepal Width')

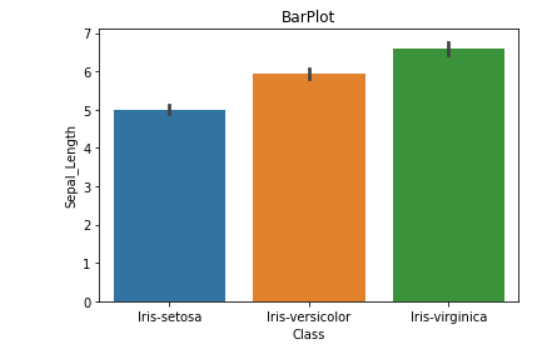
plt.ylabel('Frequency')

plt.show()



sns.barplot(iris['Class'],iris['Sepal\_Length'])

plt.title("BarPlot");



**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])   # Create a 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                  # Change an element of the array

print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array

print("\n shape of b:",b.shape)                     # Prints "(2, 3)"

print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"

a = np.zeros((3,3))   # Create an array of all zeros

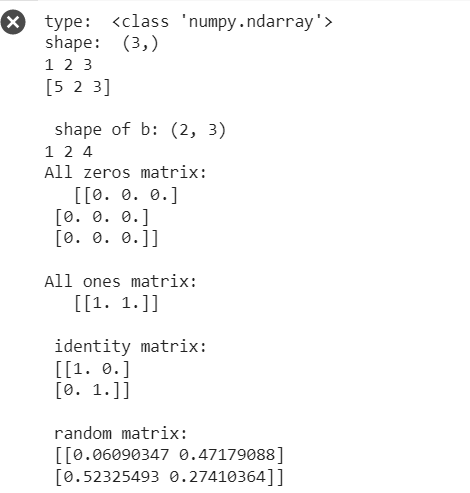
print("All zeros matrix:\n  " ,a)              # Prints "[[ 0.  0.]

b = np.ones((1,2))    # Create an array of all ones

print("\nAll ones matrix:\n  " ,b)              # Prints "[[ 1.  1.]]"

d = np.eye(2)         # Create a 2x2 identity matrix

print("\n identity matrix: \n",d)              # Prints "[[ 1.  0.]



**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)\*100)



**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:**

# Linear Regression using custom list

import numpy as np

import matplotlib.pyplot as plt

def estimate\_coef(x, y):

  n = np.size(x)

  m\_x = np.mean(x)

  m\_y = np.mean(y)

  SS\_xy = np.sum(y\*x) - n\*m\_y\*m\_x

  SS\_xx = np.sum(x\*x) - n\*m\_x\*m\_x

  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):

  plt.scatter(x, y, color = "m", marker = "o", s = 30)

  y\_pred = b[0] + b[1]\*x

  plt.plot(x, y\_pred, color = "g")

  plt.xlabel('x')

  plt.ylabel('y')

  plt.show()

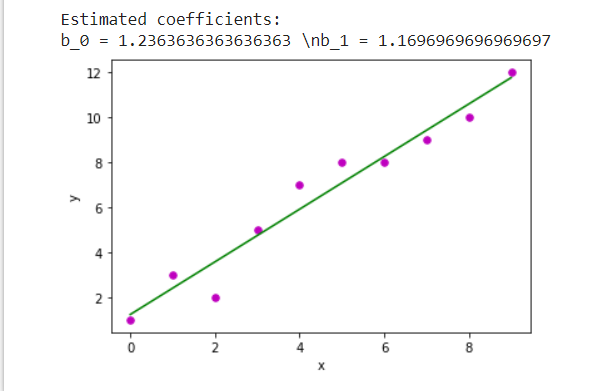
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

b = estimate\_coef(x, y)

print("Estimated coefficients:\nb\_0 = {} \\nb\_1 = {}".format(b[0], b[1]))

plot\_regression\_line(x, y, b)



# 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)

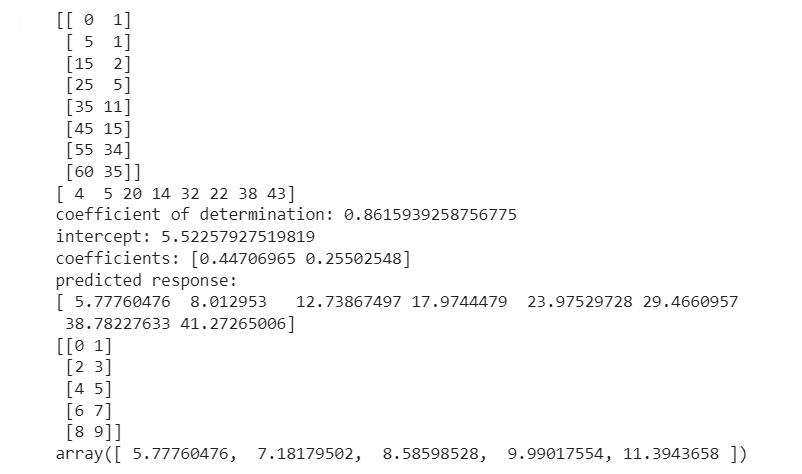
print(f"predicted response:\n{y\_pred}")

x\_new = np.arange(10).reshape((-1, 2))

print(x\_new)

y\_new = model.predict(x\_new)

y\_new



**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)

clf = DecisionTreeClassifier(max\_depth=2, random\_state=0)

clf.fit(X\_train, Y\_train)

fn = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

cn = ['setosa', 'versicolor', 'virginica']

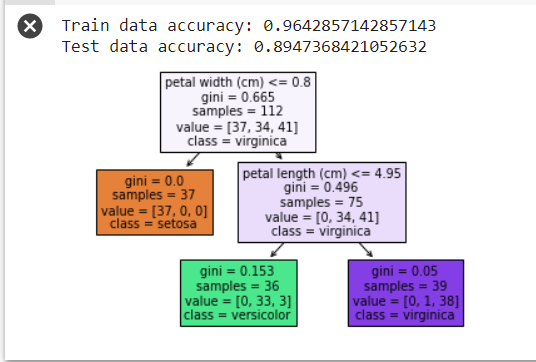
tree.plot\_tree(clf, feature\_names=fn, class\_names=cn, filled=True)

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()



**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

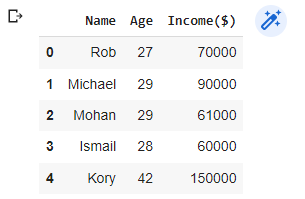
import pandas as pd

from matplotlib import pyplot as plt

%matplotlib inline

df = pd.read\_csv('income.csv')

df.head()



scaler = MinMaxScaler()

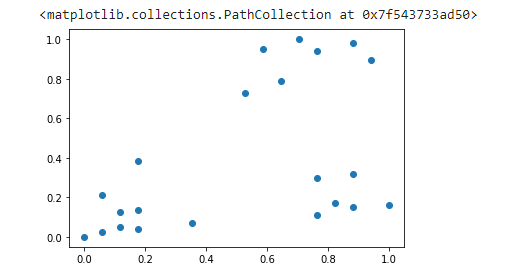
scaler.fit(df[['Income($)']])

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)

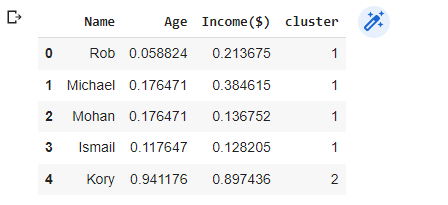
y\_predicted = km.fit\_predict(df[['Age', 'Income($)']])

y\_predicted

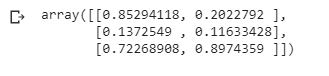


df['cluster'] = y\_predicted

df.head()



km.cluster\_centers\_



df1 = df[df.cluster==0]

df2 = df[df.cluster==1]

df3 = df[df.cluster==2]

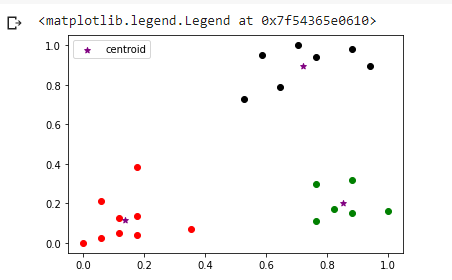
plt.scatter(df1.Age, df1['Income($)'], color = 'green')

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([])

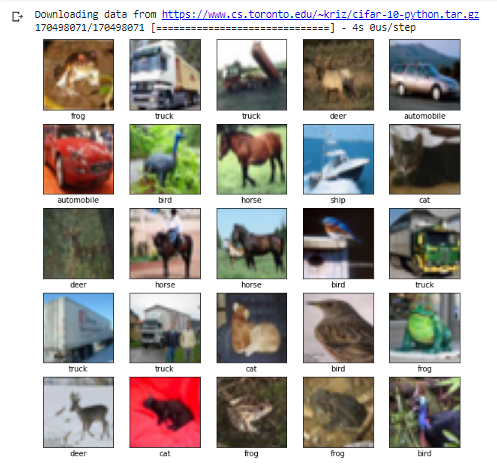
    plt.yticks([])

    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'))

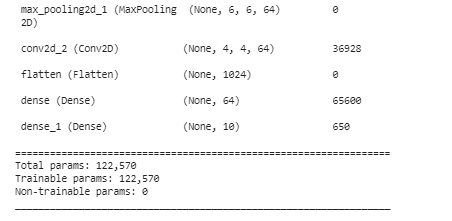
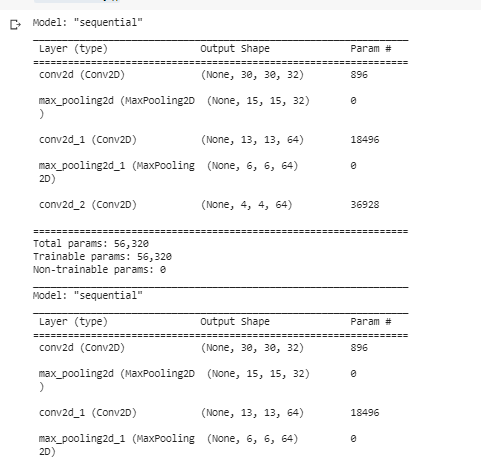
model.summary()

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(10))

model.summary()



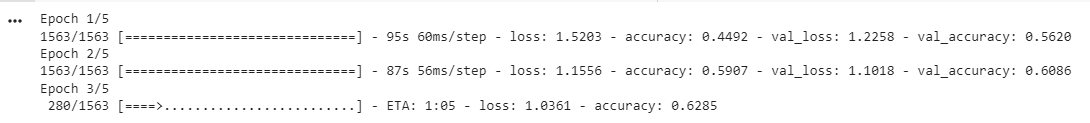
model.compile(optimizer='adam',

              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))



**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">

<head>

<meta charset="utf-8"/>

<title>

Amal Jyothi College of Engineering  
 </title>

<meta content="width=device-width, initial-scale=1" name="viewport"/>

<script type="text/javascript">

<!--

if (screen.width <= 699) {

document.location = "[https://m.ajce.in](https://m.ajce.in/)";

}

</script>

<!--[if lte IE 8]><script src="assets/js/ie/html5shiv.js"></script><![endif]-->

<link href="assets/css/main.css" rel="stylesheet"/>

<!--Bootstrap Stylesheet [ REQUIRED ]-->

<link href="css/bootstrap.css" rel="stylesheet"/>

<!--Nifty Stylesheet [ REQUIRED ]-->

<link href="css/nifty.css" rel="stylesheet"/>

<!--Animate.css [ OPTIONAL ]-->

<link href="css/animate.min.css" rel="stylesheet"/>

<link href="ajce.ico" rel="icon" type="image/ico"/>

<!--[if lte IE 8]><link rel="stylesheet" href="assets/css/ie8.css" /><![endif]-->

<!--[if lte IE 9]><link rel="stylesheet" href="assets/css/ie9.css" /><![endif]-->

<link href="../ajce.ico" rel="icon" type="image/ico"/>

<style>

.alert-title a{

border-bottom:0px;

}

</style>

</head>

<!--TIPS-->

<!--You may remove all ID or Class names which contain "demo-", they are only used for demonstration. -->

<body>

<script>

setTimeout(function(){

window.location.href = '<https://ajce.in/home/index.html>';

}, 10000);

</script>

<div class="effect aside-float aside-bright mainnav-lg" id="container">

</div>

<div id="wrapper">

<div id="bg">

</div>

<div id="overlay">

</div>

<div id="main">

<!-- Header -->

<header id="header">

<img alt="" height="100" src="300x300png.png" style="vertical-align:middle" width="100"/>

<h1>

<a href="home/index.html">

Amal Jyothi College of Engineering

</a>

</h1>

<!-- <h1><font face="Constantia" color="white"><a href="home/index.html">AMAL JYOTHI COLLEGE OF ENGINEERING</font></h1> -->

<!--<h2><a href="home/accreditations.html">ECE and EEE are re-accredited by NBA for three years till 2020</a></h2>-->

<!--<p><b>KERALA'S LARGEST INFRASTRUCTURE FOR ENGINEERING EDUCATION WITH NAAC 'A' & NBA ACCREDITATION</b></p>-->

<p>

<b>

KERALA'S LARGEST INFRASTRUCTURE FOR ENGINEERING EDUCATION WITH 6 NBA ACCREDITED PROGRAMS

</b>

</p>

<nav>

<ul>

<li>

<a class="icon fa fa-university" href="home/index.html">

HOME

</a>

</li>

<li>

<a class="icon fa fas fa-cog" href="home/btechadmissions.html">

B TECH

</a>

</li>

<li>

<a class="icon fa fas fa-cogs" href="home/mtechadmissions.html">

M TECH

</a>

</li>

<li>

<a class="icon fa fas fa-code" href="home/mcaadmissions.html">

M C A

</a>

</li>

<!--===================================================-->

<!-- END OF CONTAINER -->

<script>

window.onload = function() { document.body.className = ''; }

window.ontouchmove = function() { return false; }

window.onorientationchange = function() { document.body.scrollTop = 0; }

</script>

<!--JAVASCRIPT-->

<!--=================================================-->

<!--jQuery [ REQUIRED ]-->

<script src="js/jquery.min.js">

</script>

<!--BootstrapJS [ RECOMMENDED ]-->

<script src="js/bootstrap.min.js">

</script>

<!--NiftyJS [ RECOMMENDED ] -->

<script src="js/nifty.min.js">

</script>

<script src="js/demo/nifty-demo.min.js">

</script>

<script src="js/demo/ui-alerts.js">

</script>

</body>

</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

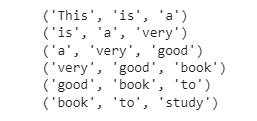
from nltk.util import ngrams

text = "This is a very good book to study";

Ngrams = ngrams(sequence = nltk.wordpunct\_tokenize(text), n=3)

for grams in Ngrams:

  print(grams)



**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', ','], ['How', 'are', 'you', '?']])



import nltk

from nltk.tag import untag

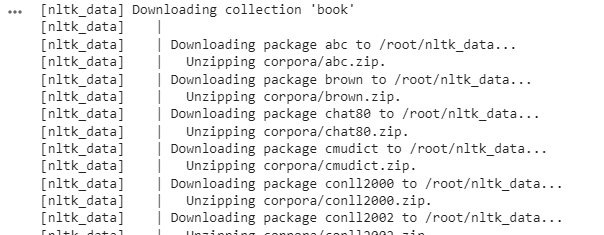
untag([('Tutorials', 'NN'), ('Point', 'NN')])

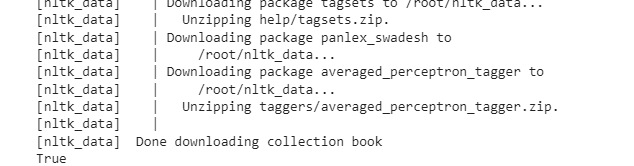


import nltk

# import all the resources for Natural Language Processing with Python

nltk.download("book")





#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)



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)



**Result:**

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

**Program No: 15**

**Aim:**

Data pre-processing with NLTK

1. Counting Tags
2. Bigrams
3. Trigrams
4. Stop Words
5. 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')

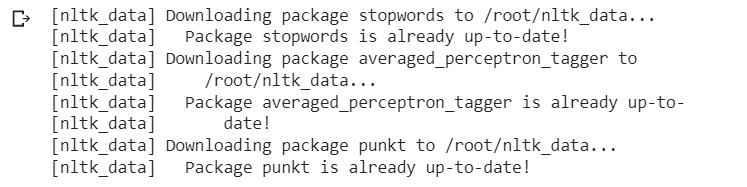
nltk.download('punkt')

import pandas as pd

import unicodedata

import numpy as np

import string



from collections import Counter

import nltk

text = "Guru99 is one of the best sites to learn WEB, SAP, Ethical Hacking and much more online."

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)



import nltk

text = "Guru99 is a totally new kind of learning experience."

Tokens = nltk.word\_tokenize(text)

output = list(nltk.bigrams(Tokens))

print(output)



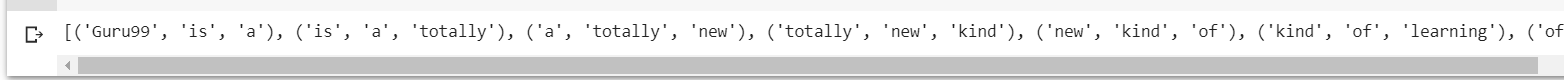
import nltk

text = "Guru99 is a totally new kind of learning experience."

Tokens = nltk.word\_tokenize(text)

output = list(nltk.trigrams(Tokens))

print(output)



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:

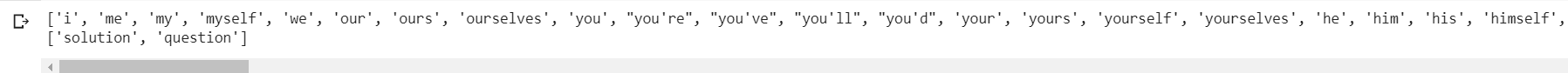
        if token not in en\_stopwords:

            result.append(token)

    return result

text = "this is the only solution of that question".split()

remove\_stopwords(text)



from nltk.stem import PorterStemmer

from nltk.tokenize import word\_tokenize

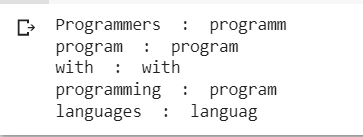
ps = PorterStemmer()

sentence = "Programmers program with programming languages"

words = word\_tokenize(sentence)

for w in words:

    print(w, " : ", ps.stem(w))



**Result:**

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