Data Analysis Using Python

CERTIFICATE

This is to certify that the practical/term work carried out in the subject of DAUP and recorded in this journal is bonafide work of Miss/Mr Ananya Vivek Kulkarni Roll No.: IT006 Identity No.: 21ECUOS146 of B. Tech. semester V in the branch of INFORMATION TECHNOLOGY, during the academic year 2023-2024.

Staff in-charge Head of the department

Date Date

**Lab Manual**

**Data Analysis Using Python**

**B.Tech. (IT)**

**SEM-VI**



**Information Technology Department**

**Faculty of Technology**

**Dharmsinh Desai University, Nadiad**

**NAME: Ananya Vivek Kulkarni**

**E-Gov ID.: 21ECUOS146**

**BATCH : H1**

**ROLL NO. : IT006**

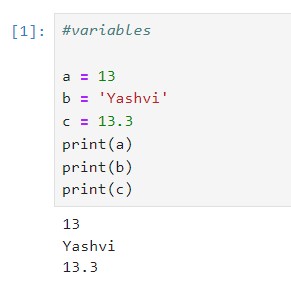
**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Title** | **Sign** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

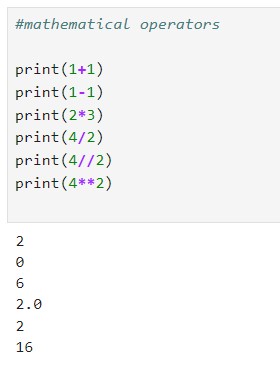
# EXPERIMENT 1

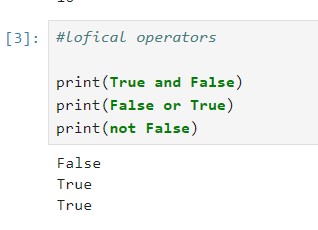
**Aim:** Introduction to Python Programming for Data Analytics.

1. Variables

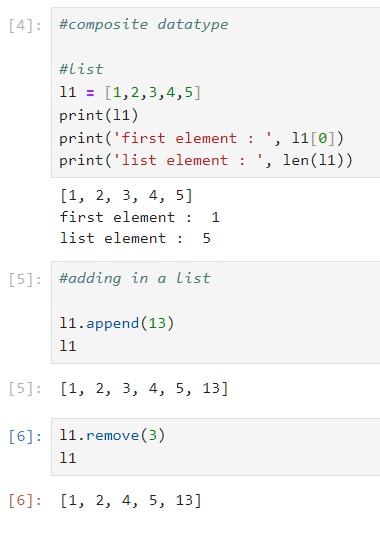


1. Mathematical and Logical Operators

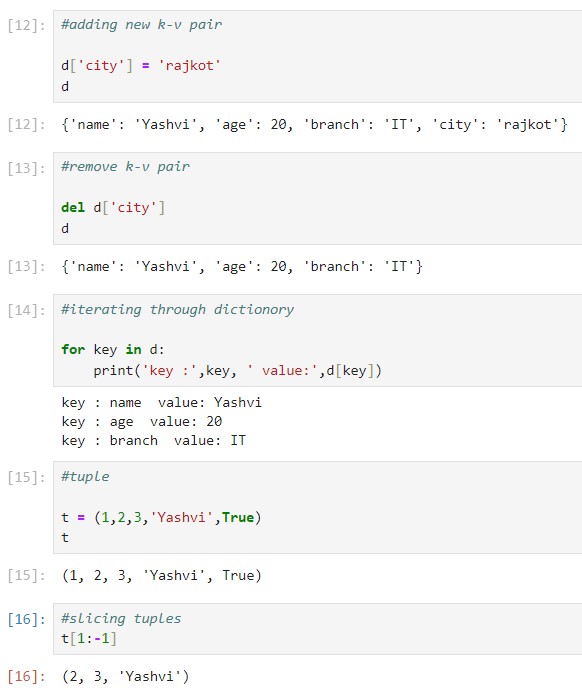




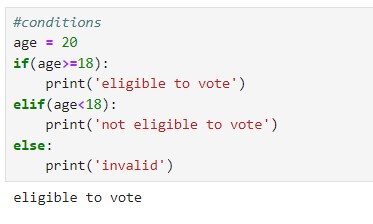
1. Composite Datatype



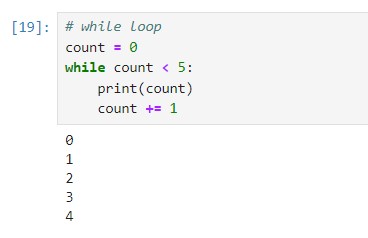


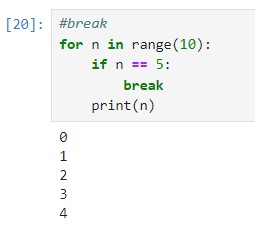


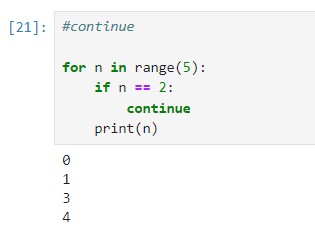
1. Conditions



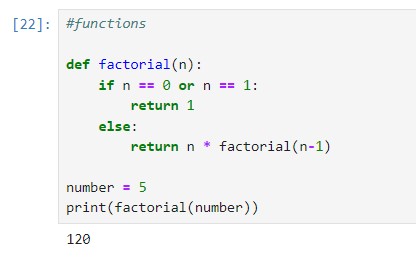
1. Loops





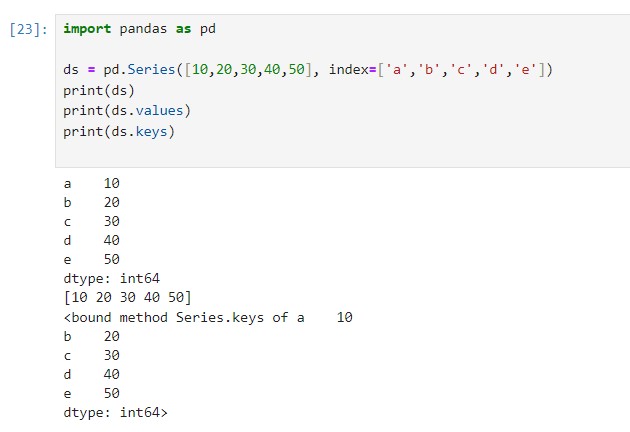


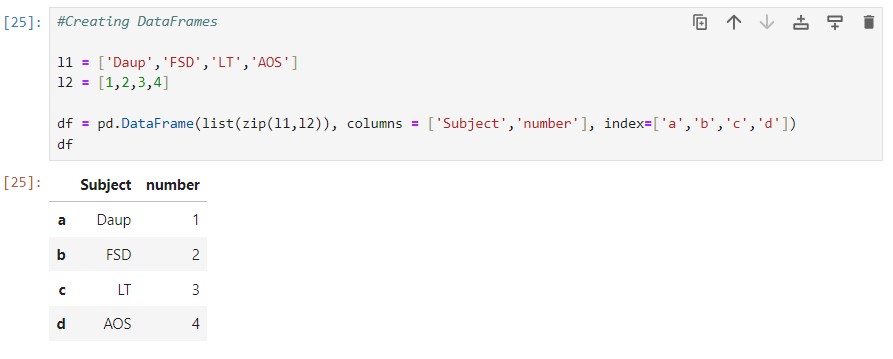
1. Functions

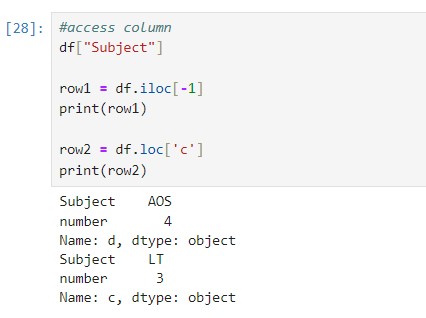


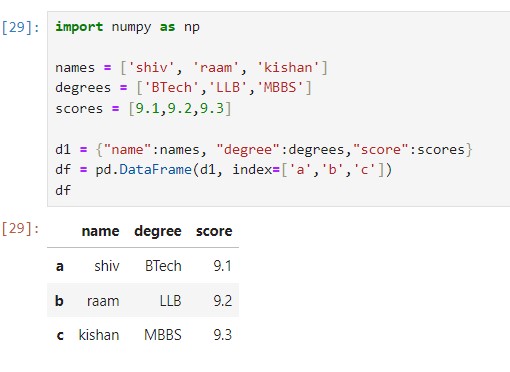
# EXPERIMENT-2

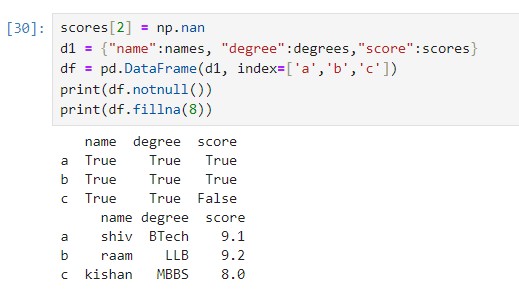
**Aim:** To perform creating and accessing Data Vector, Maths Operations, DataFrame creation, reading & writing, Handling DataFrame using Pandas and Numpy.

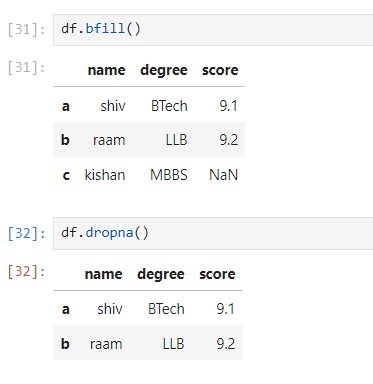


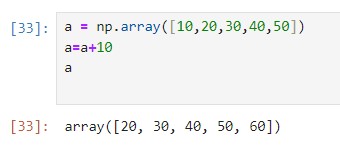


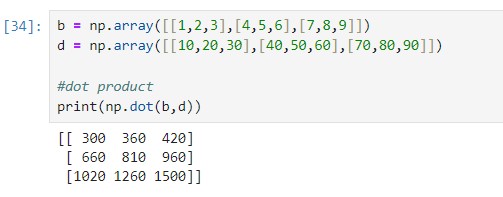


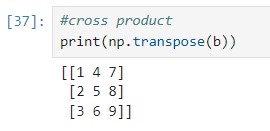


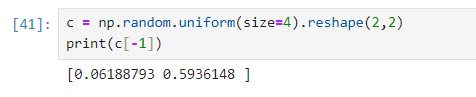


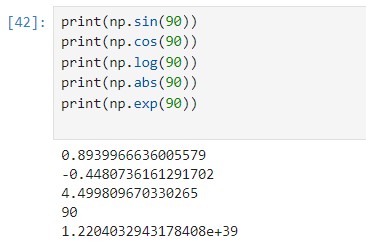






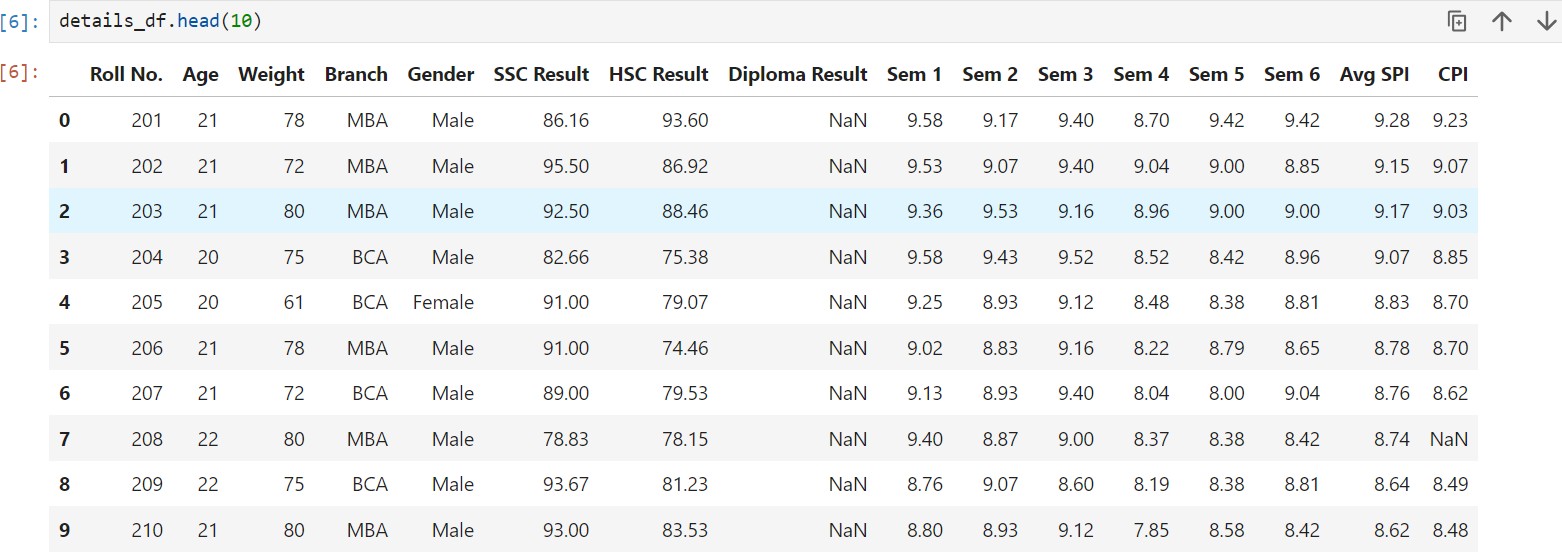


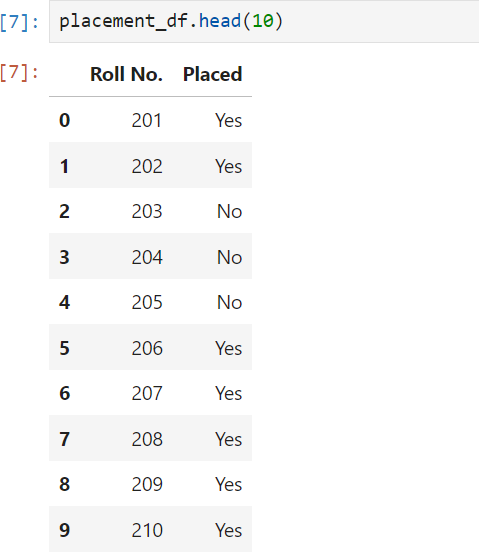


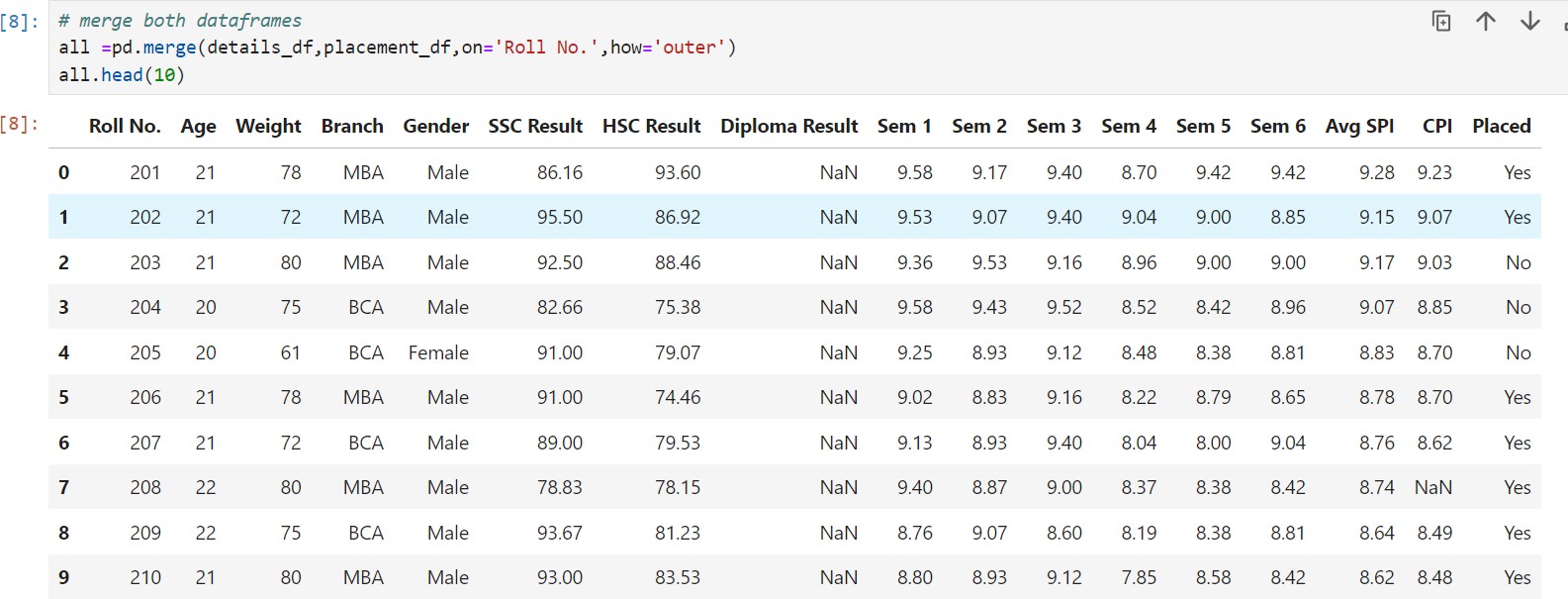


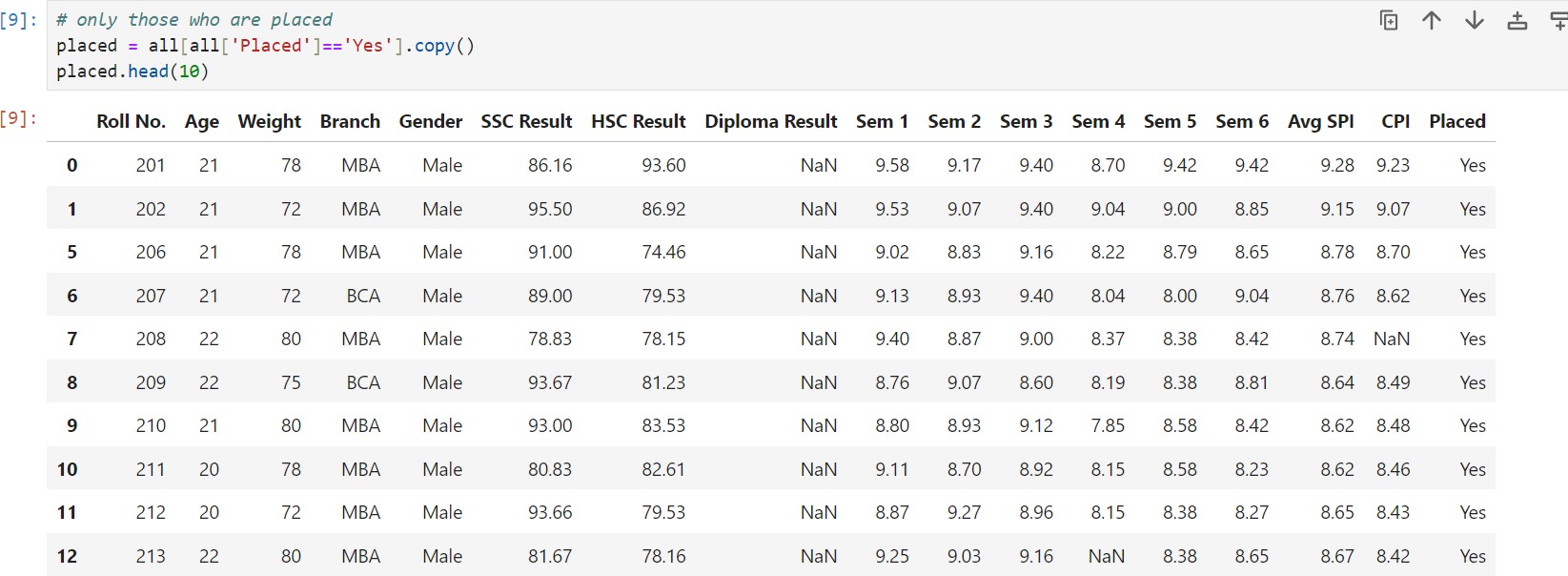
# EXPERIMENT-4

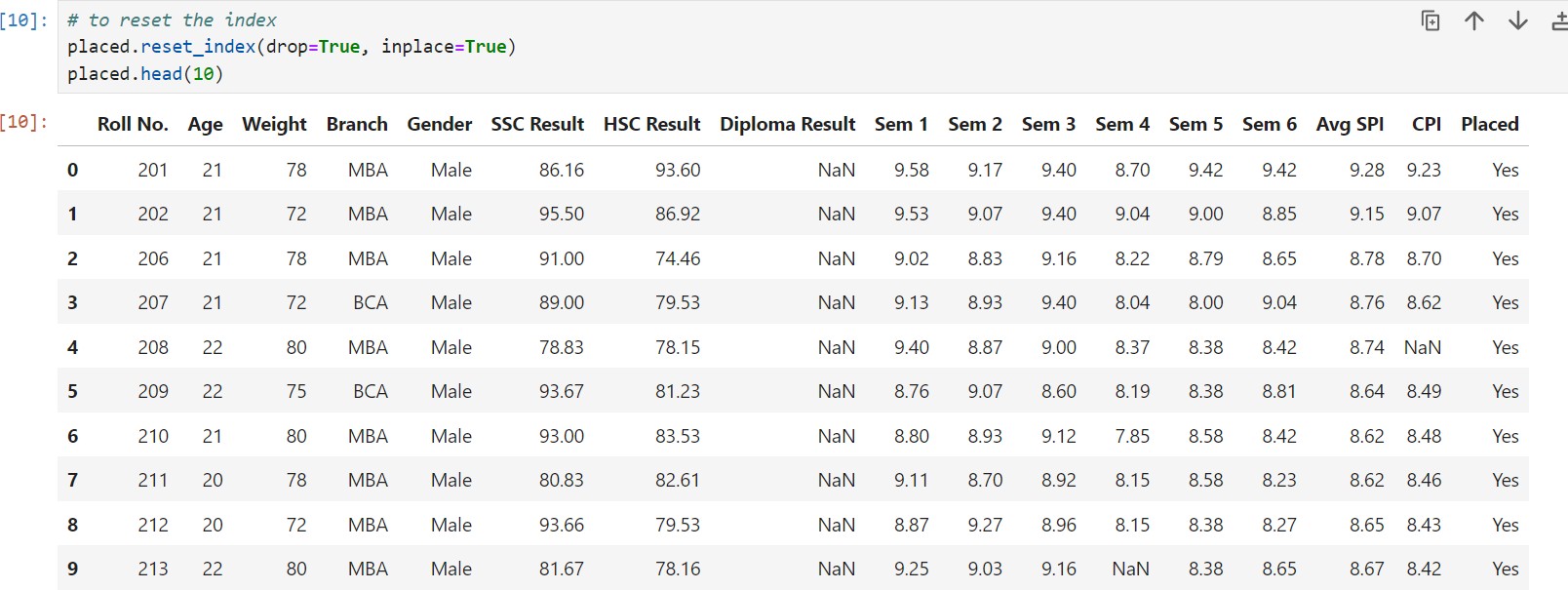
Aim: To perform Data Visualisation and plotting techniques like Lineplot, Barchart, Piechart, Boxplot using Matplotlib libraries.



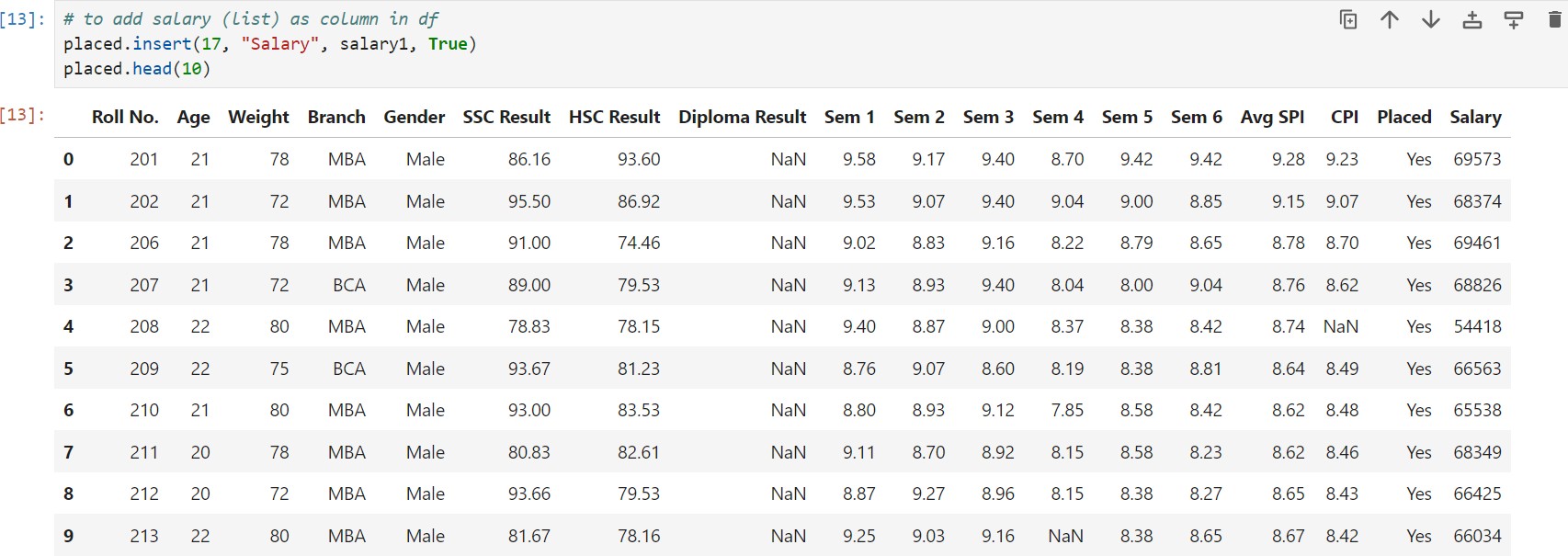


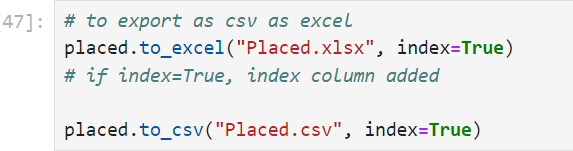


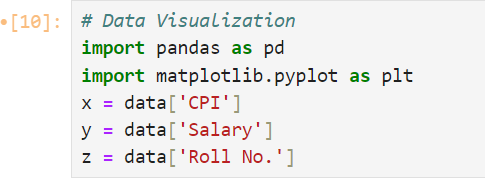


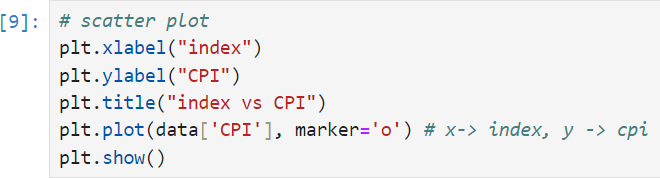


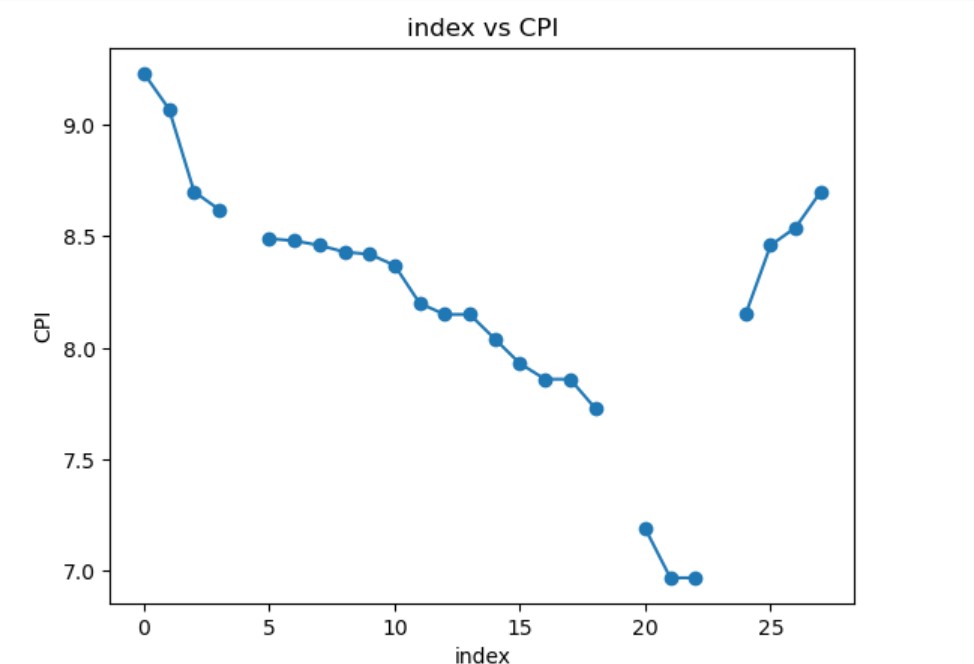


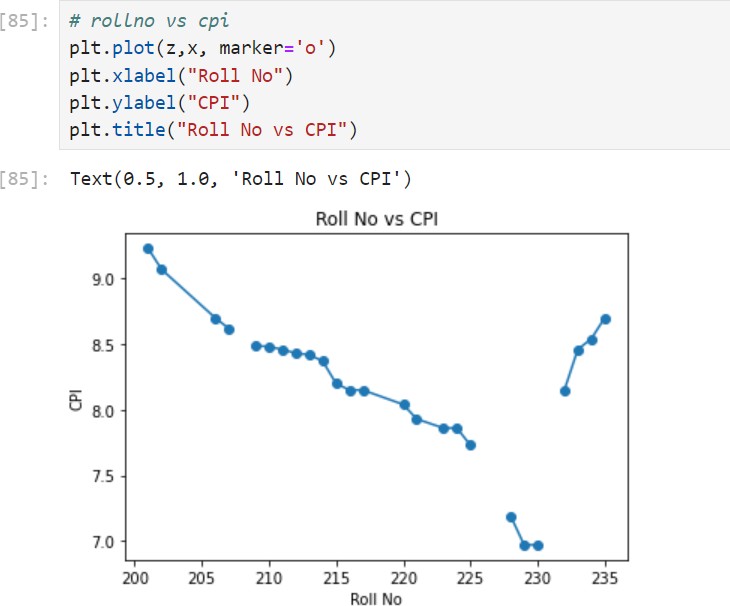


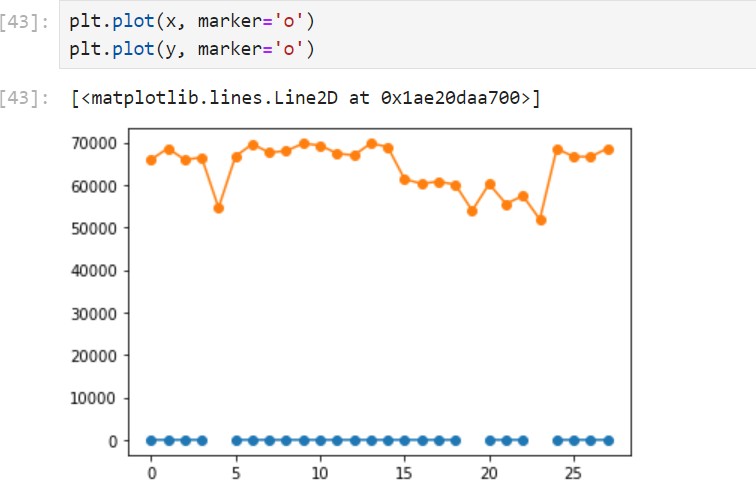


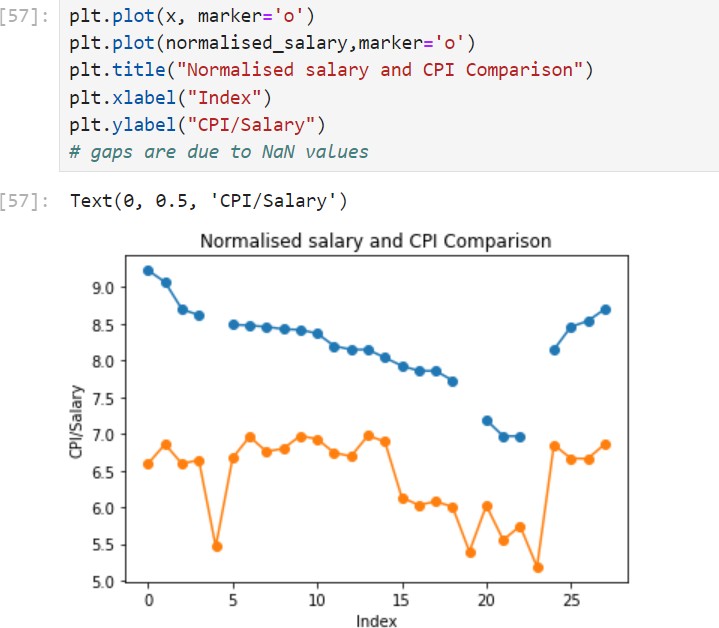
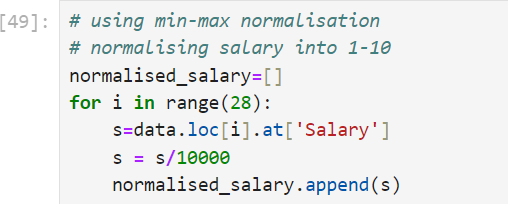


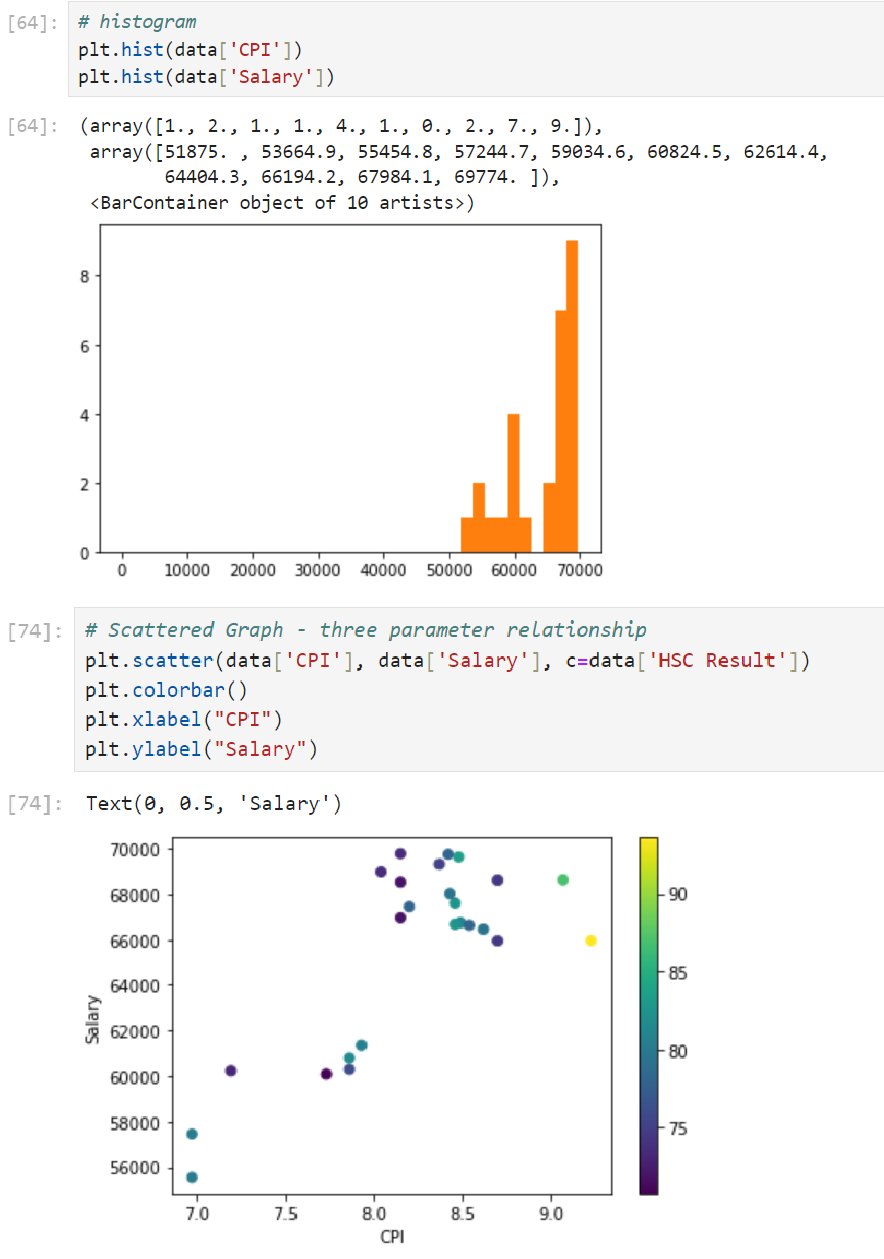


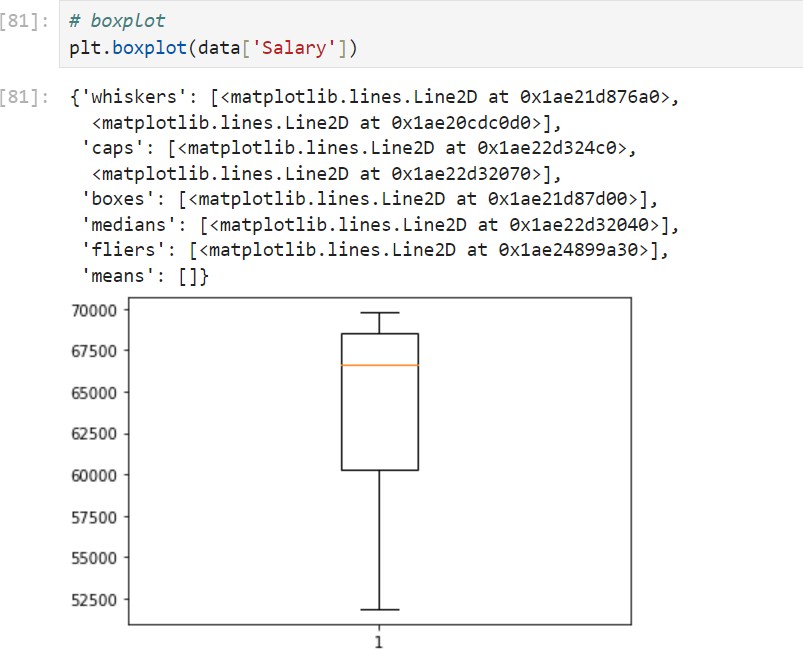


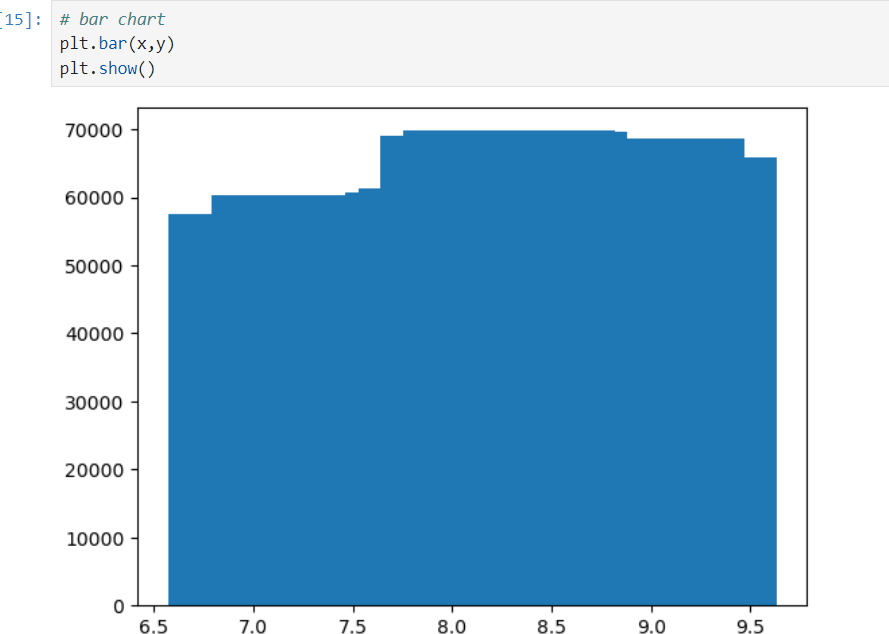


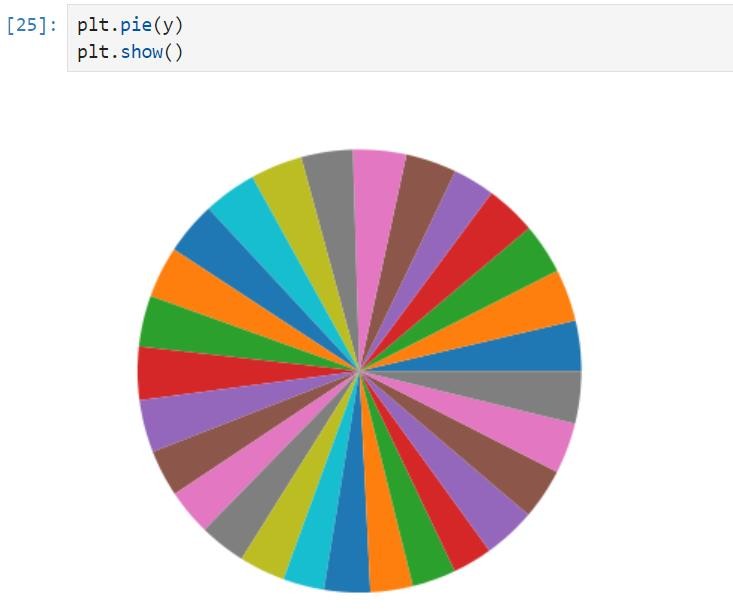






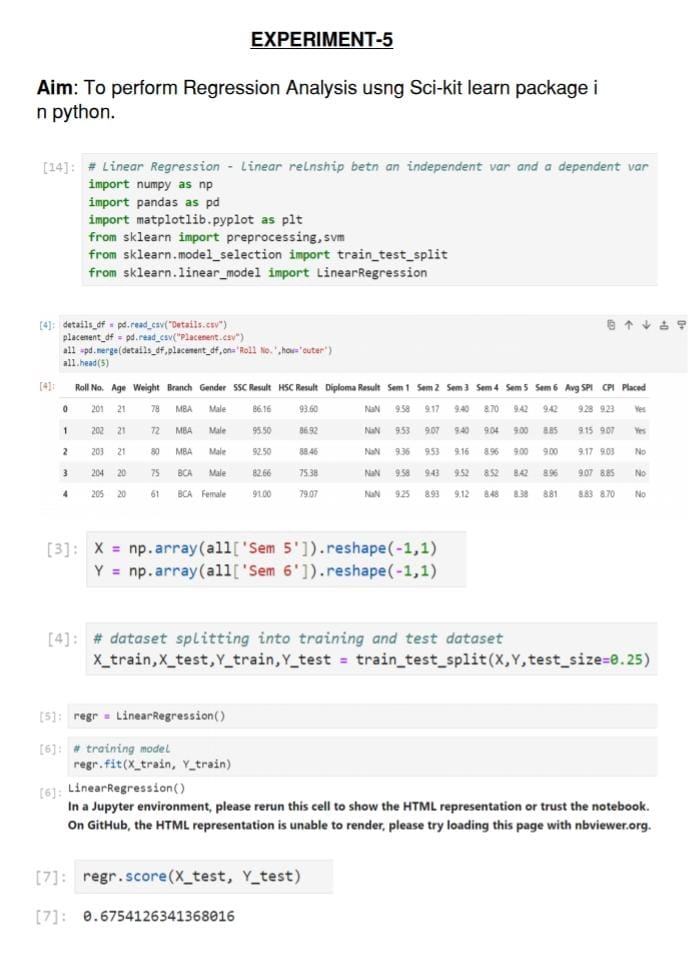






# EXPERIMENT-5

Aim: To perform Regression Analysis using Sci-kit learn package in Python







# EXPERIMENT-6

Aim: To perform Decision Tree Classification (DCT) using sklearn package in python

from sklearn.datasets import load\_iris  
from sklearn .tree import DecisionTreeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, confusion\_matrix  
import pydotplus  
from IPython.display import Image  
from sklearn.tree import export\_graphviz  
from io import StringIO  
import numpy as np  
import pandas as pd

iris=load\_iris()

print(iris)

{'data': array([[5.1, 3.5, 1.4, 0.2],  
 [4.9, 3. , 1.4, 0.2],  
 [4.7, 3.2, 1.3, 0.2],  
 [4.6, 3.1, 1.5, 0.2],  
 [5. , 3.6, 1.4, 0.2],  
 [5.4, 3.9, 1.7, 0.4],  
 [4.6, 3.4, 1.4, 0.3],  
 [5. , 3.4, 1.5, 0.2],  
 [4.4, 2.9, 1.4, 0.2],  
 [4.9, 3.1, 1.5, 0.1],  
 [5.4, 3.7, 1.5, 0.2],  
 [4.8, 3.4, 1.6, 0.2],  
 [4.8, 3. , 1.4, 0.1],  
 [4.3, 3. , 1.1, 0.1],  
 [5.8, 4. , 1.2, 0.2],  
 [5.7, 4.4, 1.5, 0.4],  
 [5.4, 3.9, 1.3, 0.4],  
 [5.1, 3.5, 1.4, 0.3],  
 [5.7, 3.8, 1.7, 0.3],  
 [5.1, 3.8, 1.5, 0.3],  
 [5.4, 3.4, 1.7, 0.2],  
 [5.1, 3.7, 1.5, 0.4],  
 [4.6, 3.6, 1. , 0.2],  
 [5.1, 3.3, 1.7, 0.5],  
 [4.8, 3.4, 1.9, 0.2],  
 [5. , 3. , 1.6, 0.2],  
 [5. , 3.4, 1.6, 0.4],  
 [5.2, 3.5, 1.5, 0.2],  
 [5.2, 3.4, 1.4, 0.2],  
 [4.7, 3.2, 1.6, 0.2],  
 [4.8, 3.1, 1.6, 0.2],  
 [5.4, 3.4, 1.5, 0.4],  
 [5.2, 4.1, 1.5, 0.1],  
 [5.5, 4.2, 1.4, 0.2],  
 [4.9, 3.1, 1.5, 0.2],  
 [5. , 3.2, 1.2, 0.2],  
 [5.5, 3.5, 1.3, 0.2],  
 [4.9, 3.6, 1.4, 0.1],  
 [4.4, 3. , 1.3, 0.2],  
 [5.1, 3.4, 1.5, 0.2],  
 [5. , 3.5, 1.3, 0.3],  
 [4.5, 2.3, 1.3, 0.3],  
 [4.4, 3.2, 1.3, 0.2],  
 [5. , 3.5, 1.6, 0.6],  
 [5.1, 3.8, 1.9, 0.4],  
 [4.8, 3. , 1.4, 0.3],  
 [5.1, 3.8, 1.6, 0.2],  
 [4.6, 3.2, 1.4, 0.2],  
 [5.3, 3.7, 1.5, 0.2],  
 [5. , 3.3, 1.4, 0.2],  
 [7. , 3.2, 4.7, 1.4],  
 [6.4, 3.2, 4.5, 1.5],  
 [6.9, 3.1, 4.9, 1.5],  
 [5.5, 2.3, 4. , 1.3],  
 [6.5, 2.8, 4.6, 1.5],  
 [5.7, 2.8, 4.5, 1.3],  
 [6.3, 3.3, 4.7, 1.6],  
 [4.9, 2.4, 3.3, 1. ],  
 [6.6, 2.9, 4.6, 1.3],  
 [5.2, 2.7, 3.9, 1.4],  
 [5. , 2. , 3.5, 1. ],  
 [5.9, 3. , 4.2, 1.5],  
 [6. , 2.2, 4. , 1. ],  
 [6.1, 2.9, 4.7, 1.4],  
 [5.6, 2.9, 3.6, 1.3],  
 [6.7, 3.1, 4.4, 1.4],  
 [5.6, 3. , 4.5, 1.5],  
 [5.8, 2.7, 4.1, 1. ],  
 [6.2, 2.2, 4.5, 1.5],  
 [5.6, 2.5, 3.9, 1.1],  
 [5.9, 3.2, 4.8, 1.8],  
 [6.1, 2.8, 4. , 1.3],  
 [6.3, 2.5, 4.9, 1.5],  
 [6.1, 2.8, 4.7, 1.2],  
 [6.4, 2.9, 4.3, 1.3],  
 [6.6, 3. , 4.4, 1.4],  
 [6.8, 2.8, 4.8, 1.4],  
 [6.7, 3. , 5. , 1.7],  
 [6. , 2.9, 4.5, 1.5],  
 [5.7, 2.6, 3.5, 1. ],  
 [5.5, 2.4, 3.8, 1.1],  
 [5.5, 2.4, 3.7, 1. ],  
 [5.8, 2.7, 3.9, 1.2],  
 [6. , 2.7, 5.1, 1.6],  
 [5.4, 3. , 4.5, 1.5],  
 [6. , 3.4, 4.5, 1.6],  
 [6.7, 3.1, 4.7, 1.5],  
 [6.3, 2.3, 4.4, 1.3],  
 [5.6, 3. , 4.1, 1.3],  
 [5.5, 2.5, 4. , 1.3],  
 [5.5, 2.6, 4.4, 1.2],  
 [6.1, 3. , 4.6, 1.4],  
 [5.8, 2.6, 4. , 1.2],  
 [5. , 2.3, 3.3, 1. ],  
 [5.6, 2.7, 4.2, 1.3],  
 [5.7, 3. , 4.2, 1.2],  
 [5.7, 2.9, 4.2, 1.3],  
 [6.2, 2.9, 4.3, 1.3],  
 [5.1, 2.5, 3. , 1.1],  
 [5.7, 2.8, 4.1, 1.3],  
 [6.3, 3.3, 6. , 2.5],  
 [5.8, 2.7, 5.1, 1.9],  
 [7.1, 3. , 5.9, 2.1],  
 [6.3, 2.9, 5.6, 1.8],  
 [6.5, 3. , 5.8, 2.2],  
 [7.6, 3. , 6.6, 2.1],  
 [4.9, 2.5, 4.5, 1.7],  
 [7.3, 2.9, 6.3, 1.8],  
 [6.7, 2.5, 5.8, 1.8],  
 [7.2, 3.6, 6.1, 2.5],  
 [6.5, 3.2, 5.1, 2. ],  
 [6.4, 2.7, 5.3, 1.9],  
 [6.8, 3. , 5.5, 2.1],  
 [5.7, 2.5, 5. , 2. ],  
 [5.8, 2.8, 5.1, 2.4],  
 [6.4, 3.2, 5.3, 2.3],  
 [6.5, 3. , 5.5, 1.8],  
 [7.7, 3.8, 6.7, 2.2],  
 [7.7, 2.6, 6.9, 2.3],  
 [6. , 2.2, 5. , 1.5],  
 [6.9, 3.2, 5.7, 2.3],  
 [5.6, 2.8, 4.9, 2. ],  
 [7.7, 2.8, 6.7, 2. ],  
 [6.3, 2.7, 4.9, 1.8],  
 [6.7, 3.3, 5.7, 2.1],  
 [7.2, 3.2, 6. , 1.8],  
 [6.2, 2.8, 4.8, 1.8],  
 [6.1, 3. , 4.9, 1.8],  
 [6.4, 2.8, 5.6, 2.1],  
 [7.2, 3. , 5.8, 1.6],  
 [7.4, 2.8, 6.1, 1.9],  
 [7.9, 3.8, 6.4, 2. ],  
 [6.4, 2.8, 5.6, 2.2],  
 [6.3, 2.8, 5.1, 1.5],  
 [6.1, 2.6, 5.6, 1.4],  
 [7.7, 3. , 6.1, 2.3],  
 [6.3, 3.4, 5.6, 2.4],  
 [6.4, 3.1, 5.5, 1.8],  
 [6. , 3. , 4.8, 1.8],  
 [6.9, 3.1, 5.4, 2.1],  
 [6.7, 3.1, 5.6, 2.4],  
 [6.9, 3.1, 5.1, 2.3],  
 [5.8, 2.7, 5.1, 1.9],  
 [6.8, 3.2, 5.9, 2.3],  
 [6.7, 3.3, 5.7, 2.5],  
 [6.7, 3. , 5.2, 2.3],  
 [6.3, 2.5, 5. , 1.9],  
 [6.5, 3. , 5.2, 2. ],  
 [6.2, 3.4, 5.4, 2.3],  
 [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,  
 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,  
 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'target\_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'), 'DESCR': '.. \_iris\_dataset:\n\nIris plants dataset\n--------------------\n\n\*\*Data Set Characteristics:\*\*\n\n:Number of Instances: 150 (50 in each of three classes)\n:Number of Attributes: 4 numeric, predictive attributes and the class\n:Attribute Information:\n - sepal length in cm\n - sepal width in cm\n - petal length in cm\n - petal width in cm\n - class:\n - Iris-Setosa\n - Iris-Versicolour\n - Iris-Virginica\n\n:Summary Statistics:\n\n============== ==== ==== ======= ===== ====================\n Min Max Mean SD Class Correlation\n============== ==== ==== ======= ===== ====================\nsepal length: 4.3 7.9 5.84 0.83 0.7826\nsepal width: 2.0 4.4 3.05 0.43 -0.4194\npetal length: 1.0 6.9 3.76 1.76 0.9490 (high!)\npetal width: 0.1 2.5 1.20 0.76 0.9565 (high!)\n============== ==== ==== ======= ===== ====================\n\n:Missing Attribute Values: None\n:Class Distribution: 33.3% for each of 3 classes.\n:Creator: R.A. Fisher\n:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n:Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning Repository, which has two wrong data points.\n\nThis is perhaps the best known database to be found in the\npattern recognition literature. Fisher\'s paper is a classic in the field and\nis referenced frequently to this day. (See Duda & Hart, for example.) The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of iris plant. One class is linearly separable from the other 2; the\nlatter are NOT linearly separable from each other.\n\n|details-start|\n\*\*References\*\*\n|details-split|\n\n- Fisher, R.A. "The use of multiple measurements in taxonomic problems"\n Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n Mathematical Statistics" (John Wiley, NY, 1950).\n- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.\n- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n Structure and Classification Rule for Recognition in Partially Exposed\n Environments". IEEE Transactions on Pattern Analysis and Machine\n Intelligence, Vol. PAMI-2, No. 1, 67-71.\n- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions\n on Information Theory, May 1972, 431-433.\n- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al"s AUTOCLASS II\n conceptual clustering system finds 3 classes in the data.\n- Many, many more ...\n\n|details-end|\n', 'feature\_names': ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data\_module': 'sklearn.datasets.data'}

X= iris.data  
y= iris.target

X\_train,X\_test, y\_train, y\_test =train\_test\_split(X,y,test\_size=0.3,)

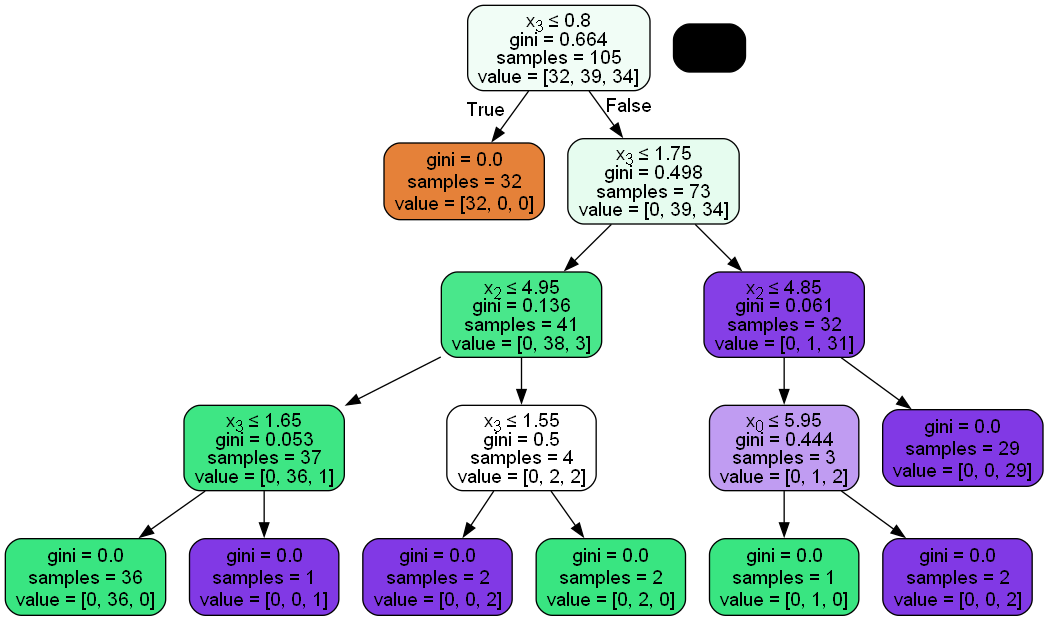
dct = DecisionTreeClassifier()

dct.fit(X\_train, y\_train)  
y\_hat = dct.predict(X\_test)

print(accuracy\_score(y\_test,y\_hat))  
print(confusion\_matrix(y\_test,y\_hat))

0.9777777777777777  
[[18 0 0]  
 [ 0 11 0]  
 [ 0 1 15]]

dot\_data =StringIO()  
export\_graphviz(dct, out\_file=dot\_data, filled=True,rounded=True, special\_characters=True)  
graph= pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())  
Image(graph.create\_png())



# EXPERIMENT-7

# Aim: To perform Navie Bayes and K-NN (k-nearest

# neighbor) classification using sklearn package in python

**KNN**

from sklearn.datasets import load\_iris  
from sklearn .tree import DecisionTreeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, confusion\_matrix  
from sklearn.neighbors import KNeighborsClassifier # Correct import  
from sklearn.naive\_bayes import GaussianNB # No conflict  
  
import pydotplus  
from IPython.display import Image  
from sklearn.tree import export\_graphviz  
from io import StringIO  
import numpy as np  
import pandas as pd

iris=load\_iris()

print(iris)

{'data': array([[5.1, 3.5, 1.4, 0.2],  
 [4.9, 3. , 1.4, 0.2],  
 [4.7, 3.2, 1.3, 0.2],  
 [4.6, 3.1, 1.5, 0.2],  
 [5. , 3.6, 1.4, 0.2],  
 [5.4, 3.9, 1.7, 0.4],  
 [4.6, 3.4, 1.4, 0.3],  
 [5. , 3.4, 1.5, 0.2],  
 [4.4, 2.9, 1.4, 0.2],  
 [4.9, 3.1, 1.5, 0.1],  
 [5.4, 3.7, 1.5, 0.2],  
 [4.8, 3.4, 1.6, 0.2],  
 [4.8, 3. , 1.4, 0.1],  
 [4.3, 3. , 1.1, 0.1],  
 [5.8, 4. , 1.2, 0.2],  
 [5.7, 4.4, 1.5, 0.4],  
 [5.4, 3.9, 1.3, 0.4],  
 [5.1, 3.5, 1.4, 0.3],  
 [5.7, 3.8, 1.7, 0.3],  
 [5.1, 3.8, 1.5, 0.3],  
 [5.4, 3.4, 1.7, 0.2],  
 [5.1, 3.7, 1.5, 0.4],  
 [4.6, 3.6, 1. , 0.2],  
 [5.1, 3.3, 1.7, 0.5],  
 [4.8, 3.4, 1.9, 0.2],  
 [5. , 3. , 1.6, 0.2],  
 [5. , 3.4, 1.6, 0.4],  
 [5.2, 3.5, 1.5, 0.2],  
 [5.2, 3.4, 1.4, 0.2],  
 [4.7, 3.2, 1.6, 0.2],  
 [4.8, 3.1, 1.6, 0.2],  
 [5.4, 3.4, 1.5, 0.4],  
 [5.2, 4.1, 1.5, 0.1],  
 [5.5, 4.2, 1.4, 0.2],  
 [4.9, 3.1, 1.5, 0.2],  
 [5. , 3.2, 1.2, 0.2],  
 [5.5, 3.5, 1.3, 0.2],  
 [4.9, 3.6, 1.4, 0.1],  
 [4.4, 3. , 1.3, 0.2],  
 [5.1, 3.4, 1.5, 0.2],  
 [5. , 3.5, 1.3, 0.3],  
 [4.5, 2.3, 1.3, 0.3],  
 [4.4, 3.2, 1.3, 0.2],  
 [5. , 3.5, 1.6, 0.6],  
 [5.1, 3.8, 1.9, 0.4],  
 [4.8, 3. , 1.4, 0.3],  
 [5.1, 3.8, 1.6, 0.2],  
 [4.6, 3.2, 1.4, 0.2],  
 [5.3, 3.7, 1.5, 0.2],  
 [5. , 3.3, 1.4, 0.2],  
 [7. , 3.2, 4.7, 1.4],  
 [6.4, 3.2, 4.5, 1.5],  
 [6.9, 3.1, 4.9, 1.5],  
 [5.5, 2.3, 4. , 1.3],  
 [6.5, 2.8, 4.6, 1.5],  
 [5.7, 2.8, 4.5, 1.3],  
 [6.3, 3.3, 4.7, 1.6],  
 [4.9, 2.4, 3.3, 1. ],  
 [6.6, 2.9, 4.6, 1.3],  
 [5.2, 2.7, 3.9, 1.4],  
 [5. , 2. , 3.5, 1. ],  
 [5.9, 3. , 4.2, 1.5],  
 [6. , 2.2, 4. , 1. ],  
 [6.1, 2.9, 4.7, 1.4],  
 [5.6, 2.9, 3.6, 1.3],  
 [6.7, 3.1, 4.4, 1.4],  
 [5.6, 3. , 4.5, 1.5],  
 [5.8, 2.7, 4.1, 1. ],  
 [6.2, 2.2, 4.5, 1.5],  
 [5.6, 2.5, 3.9, 1.1],  
 [5.9, 3.2, 4.8, 1.8],  
 [6.1, 2.8, 4. , 1.3],  
 [6.3, 2.5, 4.9, 1.5],  
 [6.1, 2.8, 4.7, 1.2],  
 [6.4, 2.9, 4.3, 1.3],  
 [6.6, 3. , 4.4, 1.4],  
 [6.8, 2.8, 4.8, 1.4],  
 [6.7, 3. , 5. , 1.7],  
 [6. , 2.9, 4.5, 1.5],  
 [5.7, 2.6, 3.5, 1. ],  
 [5.5, 2.4, 3.8, 1.1],  
 [5.5, 2.4, 3.7, 1. ],  
 [5.8, 2.7, 3.9, 1.2],  
 [6. , 2.7, 5.1, 1.6],  
 [5.4, 3. , 4.5, 1.5],  
 [6. , 3.4, 4.5, 1.6],  
 [6.7, 3.1, 4.7, 1.5],  
 [6.3, 2.3, 4.4, 1.3],  
 [5.6, 3. , 4.1, 1.3],  
 [5.5, 2.5, 4. , 1.3],  
 [5.5, 2.6, 4.4, 1.2],  
 [6.1, 3. , 4.6, 1.4],  
 [5.8, 2.6, 4. , 1.2],  
 [5. , 2.3, 3.3, 1. ],  
 [5.6, 2.7, 4.2, 1.3],  
 [5.7, 3. , 4.2, 1.2],  
 [5.7, 2.9, 4.2, 1.3],  
 [6.2, 2.9, 4.3, 1.3],  
 [5.1, 2.5, 3. , 1.1],  
 [5.7, 2.8, 4.1, 1.3],  
 [6.3, 3.3, 6. , 2.5],  
 [5.8, 2.7, 5.1, 1.9],  
 [7.1, 3. , 5.9, 2.1],  
 [6.3, 2.9, 5.6, 1.8],  
 [6.5, 3. , 5.8, 2.2],  
 [7.6, 3. , 6.6, 2.1],  
 [4.9, 2.5, 4.5, 1.7],  
 [7.3, 2.9, 6.3, 1.8],  
 [6.7, 2.5, 5.8, 1.8],  
 [7.2, 3.6, 6.1, 2.5],  
 [6.5, 3.2, 5.1, 2. ],  
 [6.4, 2.7, 5.3, 1.9],  
 [6.8, 3. , 5.5, 2.1],  
 [5.7, 2.5, 5. , 2. ],  
 [5.8, 2.8, 5.1, 2.4],  
 [6.4, 3.2, 5.3, 2.3],  
 [6.5, 3. , 5.5, 1.8],  
 [7.7, 3.8, 6.7, 2.2],  
 [7.7, 2.6, 6.9, 2.3],  
 [6. , 2.2, 5. , 1.5],  
 [6.9, 3.2, 5.7, 2.3],  
 [5.6, 2.8, 4.9, 2. ],  
 [7.7, 2.8, 6.7, 2. ],  
 [6.3, 2.7, 4.9, 1.8],  
 [6.7, 3.3, 5.7, 2.1],  
 [7.2, 3.2, 6. , 1.8],  
 [6.2, 2.8, 4.8, 1.8],  
 [6.1, 3. , 4.9, 1.8],  
 [6.4, 2.8, 5.6, 2.1],  
 [7.2, 3. , 5.8, 1.6],  
 [7.4, 2.8, 6.1, 1.9],  
 [7.9, 3.8, 6.4, 2. ],  
 [6.4, 2.8, 5.6, 2.2],  
 [6.3, 2.8, 5.1, 1.5],  
 [6.1, 2.6, 5.6, 1.4],  
 [7.7, 3. , 6.1, 2.3],  
 [6.3, 3.4, 5.6, 2.4],  
 [6.4, 3.1, 5.5, 1.8],  
 [6. , 3. , 4.8, 1.8],  
 [6.9, 3.1, 5.4, 2.1],  
 [6.7, 3.1, 5.6, 2.4],  
 [6.9, 3.1, 5.1, 2.3],  
 [5.8, 2.7, 5.1, 1.9],  
 [6.8, 3.2, 5.9, 2.3],  
 [6.7, 3.3, 5.7, 2.5],  
 [6.7, 3. , 5.2, 2.3],  
 [6.3, 2.5, 5. , 1.9],  
 [6.5, 3. , 5.2, 2. ],  
 [6.2, 3.4, 5.4, 2.3],  
 [5.9, 3. , 5.1, 1.8]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,  
 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,  
 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'target\_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'), 'DESCR': '.. \_iris\_dataset:\n\nIris plants dataset\n--------------------\n\n\*\*Data Set Characteristics:\*\*\n\n:Number of Instances: 150 (50 in each of three classes)\n:Number of Attributes: 4 numeric, predictive attributes and the class\n:Attribute Information:\n - sepal length in cm\n - sepal width in cm\n - petal length in cm\n - petal width in cm\n - class:\n - Iris-Setosa\n - Iris-Versicolour\n - Iris-Virginica\n\n:Summary Statistics:\n\n============== ==== ==== ======= ===== ====================\n Min Max Mean SD Class Correlation\n============== ==== ==== ======= ===== ====================\nsepal length: 4.3 7.9 5.84 0.83 0.7826\nsepal width: 2.0 4.4 3.05 0.43 -0.4194\npetal length: 1.0 6.9 3.76 1.76 0.9490 (high!)\npetal width: 0.1 2.5 1.20 0.76 0.9565 (high!)\n============== ==== ==== ======= ===== ====================\n\n:Missing Attribute Values: None\n:Class Distribution: 33.3% for each of 3 classes.\n:Creator: R.A. Fisher\n:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n:Date: July, 1988\n\nThe famous Iris database, first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the same as in R, but not as in the UCI\nMachine Learning Repository, which has two wrong data points.\n\nThis is perhaps the best known database to be found in the\npattern recognition literature. Fisher\'s paper is a classic in the field and\nis referenced frequently to this day. (See Duda & Hart, for example.) The\ndata set contains 3 classes of 50 instances each, where each class refers to a\ntype of iris plant. One class is linearly separable from the other 2; the\nlatter are NOT linearly separable from each other.\n\n|details-start|\n\*\*References\*\*\n|details-split|\n\n- Fisher, R.A. "The use of multiple measurements in taxonomic problems"\n Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to\n Mathematical Statistics" (John Wiley, NY, 1950).\n- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n (Q327.D83) John Wiley & Sons. ISBN 0-471-22361-1. See page 218.\n- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System\n Structure and Classification Rule for Recognition in Partially Exposed\n Environments". IEEE Transactions on Pattern Analysis and Machine\n Intelligence, Vol. PAMI-2, No. 1, 67-71.\n- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule". IEEE Transactions\n on Information Theory, May 1972, 431-433.\n- See also: 1988 MLC Proceedings, 54-64. Cheeseman et al"s AUTOCLASS II\n conceptual clustering system finds 3 classes in the data.\n- Many, many more ...\n\n|details-end|\n', 'feature\_names': ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'iris.csv', 'data\_module': 'sklearn.datasets.data'}

X= iris.data  
y= iris.target

X\_train,X\_test, y\_train, y\_test =train\_test\_split(X,y,test\_size=0.33,stratify=y)

nb = GaussianNB()

knn = KNeighborsClassifier(n\_neighbors=5)

# Train the models (assuming you want to do that)  
knn.fit(X\_train, y\_train)  
nb.fit(X\_train, y\_train)  
  
# Make predictions and evaluate performance (consider replacing accuracy with more appropriate metrics)  
knn\_predictions = knn.predict(X\_test)  
nb\_predictions = nb.predict(X\_test)  
  
knn\_accuracy = accuracy\_score(y\_test, knn\_predictions)  
nb\_accuracy = accuracy\_score(y\_test, nb\_predictions)  
print(confusion\_matrix(y\_test,knn\_predictions))  
print(confusion\_matrix(y\_test,nb\_predictions))  
  
print("KNN Accuracy:", knn\_accuracy)  
print("GaussianNB Accuracy:", nb\_accuracy)

[[17 0 0]  
 [ 0 15 1]  
 [ 0 0 17]]  
[[17 0 0]  
 [ 0 15 1]  
 [ 0 0 17]]  
KNN Accuracy: 0.98  
GaussianNB Accuracy: 0.98

**Naïve Bayes Classification**

from sklearn.datasets import load\_iris  
from sklearn .tree import DecisionTreeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, confusion\_matrix  
from sklearn.naive\_bayes import GaussianNB # No conflict  
from sklearn.preprocessing import LabelEncoder  
import numpy as np  
import pandas as pd

weather = pd.read\_csv('weather.csv')

weather

outlook temperature humidity windy play  
0 overcast hot high False yes  
1 overcast cool normal True yes  
2 overcast mild high True yes  
3 overcast hot normal False yes  
4 rainy mild high False yes  
5 rainy cool normal False yes  
6 rainy cool normal True no  
7 rainy mild normal False yes  
8 rainy mild high True no  
9 sunny hot high False no  
10 sunny hot high True no  
11 sunny mild high False no  
12 sunny cool normal False yes  
13 sunny mild normal True yes

Numerics=LabelEncoder()

inputs=weather.drop('play',axis='columns')  
target=weather['play']  
target

0 yes  
1 yes  
2 yes  
3 yes  
4 yes  
5 yes  
6 no  
7 yes  
8 no  
9 no  
10 no  
11 no  
12 yes  
13 yes  
Name: play, dtype: object

inputs['outlook\_n']=Numerics.fit\_transform(inputs['outlook'])  
inputs['temp\_n']=Numerics.fit\_transform(inputs['temperature'])  
inputs['hum\_n']=Numerics.fit\_transform(inputs['humidity'])  
inputs['wind\_n']=Numerics.fit\_transform(inputs['windy'])  
inputs

outlook temperature humidity windy outlook\_n temp\_n hum\_n wind\_n  
0 overcast hot high False 0 1 0 0  
1 overcast cool normal True 0 0 1 1  
2 overcast mild high True 0 2 0 1  
3 overcast hot normal False 0 1 1 0  
4 rainy mild high False 1 2 0 0  
5 rainy cool normal False 1 0 1 0  
6 rainy cool normal True 1 0 1 1  
7 rainy mild normal False 1 2 1 0  
8 rainy mild high True 1 2 0 1  
9 sunny hot high False 2 1 0 0  
10 sunny hot high True 2 1 0 1  
11 sunny mild high False 2 2 0 0  
12 sunny cool normal False 2 0 1 0  
13 sunny mild normal True 2 2 1 1

inputs\_n=inputs.drop(['outlook','temperature','humidity','windy'],axis='columns')  
inputs\_n

outlook\_n temp\_n hum\_n wind\_n  
0 0 1 0 0  
1 0 0 1 1  
2 0 2 0 1  
3 0 1 1 0  
4 1 2 0 0  
5 1 0 1 0  
6 1 0 1 1  
7 1 2 1 0  
8 1 2 0 1  
9 2 1 0 0  
10 2 1 0 1  
11 2 2 0 0  
12 2 0 1 0  
13 2 2 1 1

Classifier=GaussianNB()

Classifier.fit(inputs\_n,target)

GaussianNB()

Classifier.score(inputs\_n,target)

0.9285714285714286

Classifier.predict([[0,1,1,0]])

c:\Users\anany\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but GaussianNB was fitted with feature names  
 warnings.warn(

array(['yes'], dtype='<U3')

**Multinomial Naïve Bayes Classification**

import numpy as np   
import pandas as pd   
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.metrics import confusion\_matrix  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn.pipeline import Pipeline  
from sklearn.feature\_extraction.text import CountVectorizer  
from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

data=pd.read\_csv('spam.csv')  
data

Category Message  
0 ham Go until jurong point, crazy.. Available only ...  
1 ham Ok lar... Joking wif u oni...  
2 spam Free entry in 2 a wkly comp to win FA Cup fina...  
3 ham U dun say so early hor... U c already then say...  
4 ham Nah I don't think he goes to usf, he lives aro...  
... ... ...  
5567 spam This is the 2nd time we have tried 2 contact u...  
5568 ham Will ü b going to esplanade fr home?  
5569 ham Pity, \* was in mood for that. So...any other s...  
5570 ham The guy did some bitching but I acted like i'd...  
5571 ham Rofl. Its true to its name  
  
[5572 rows x 2 columns]

data.columns

Index(['Category', 'Message'], dtype='object')

data.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5572 entries, 0 to 5571  
Data columns (total 2 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Category 5572 non-null object  
 1 Message 5572 non-null object  
dtypes: object(2)  
memory usage: 87.2+ KB

data.isna().sum()

Category 0  
Message 0  
dtype: int64

data['Spam']=data['Category'].apply(lambda x:1 if x=='spam' else 0)  
data.head(5)

Category Message Spam  
0 ham Go until jurong point, crazy.. Available only ... 0  
1 ham Ok lar... Joking wif u oni... 0  
2 spam Free entry in 2 a wkly comp to win FA Cup fina... 1  
3 ham U dun say so early hor... U c already then say... 0  
4 ham Nah I don't think he goes to usf, he lives aro... 0

X\_train,X\_test,y\_train,y\_test=train\_test\_split(data.Message,data.Spam,test\_size=0.25)

clf=Pipeline([  
 ('vectorizer',CountVectorizer()),  
 ('nb',MultinomialNB())  
])

clf.fit(X\_train,y\_train)

Pipeline(steps=[('vectorizer', CountVectorizer()), ('nb', MultinomialNB())])

emails=[  
 'Sounds great! Are you home now?',  
 'Will u meet ur dream partner soon? Is ur career off 2 a flyng start? 2 find out free, txt HORO followed by ur star sign, e. g. HORO ARIES'  
]

clf.predict(emails)

array([0, 1], dtype=int64)

clf.score(X\_test,y\_test)

0.9856424982053122

df = pd.read\_csv('spam.csv')  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['Message'], df['Category'], test\_size=0.2, random\_state=42)  
  
vectorizer = CountVectorizer()  
  
X\_train\_vectorized = vectorizer.fit\_transform(X\_train)  
  
classifier = MultinomialNB()  
classifier.fit(X\_train\_vectorized, y\_train)  
  
X\_test\_vectorized = vectorizer.transform(X\_test)  
  
# Make predictions  
predictions = classifier.predict(X\_test\_vectorized)  
  
  
# Evaluate the model  
accuracy = accuracy\_score(y\_test, predictions)  
conf\_matrix = confusion\_matrix(y\_test, predictions)  
classification\_rep = classification\_report(y\_test, predictions)  
  
print(f'Accuracy: {accuracy:.2f}')  
print('Confusion Matrix:')  
print(conf\_matrix)  
print('Classification Report:')  
print(y\_train)

Accuracy: 0.99  
Confusion Matrix:  
[[966 0]  
 [ 9 140]]  
Classification Report:  
1978 spam  
3989 ham  
3935 ham  
4078 ham  
4086 spam  
 ...   
3772 ham  
5191 ham  
5226 ham  
5390 ham  
860 ham  
Name: Category, Length: 4457, dtype: object

emails = ['Hey mohan, can we get together to watch football game tomorrow?','Upto 20% discount on parking, exclusive offer just for you. Dont miss this reawd!']  
emails\_count = vectorizer.transform(emails)  
classifier.predict(emails\_count)  
  
X\_test\_count = vectorizer.transform(X\_test)  
classifier.score(X\_test\_count,y\_test)

0.9919282511210762

# EXPERIMENT-8

Aim: To perform KMeans and DBSCAN Clustering technique using Sci-kit learn package in python

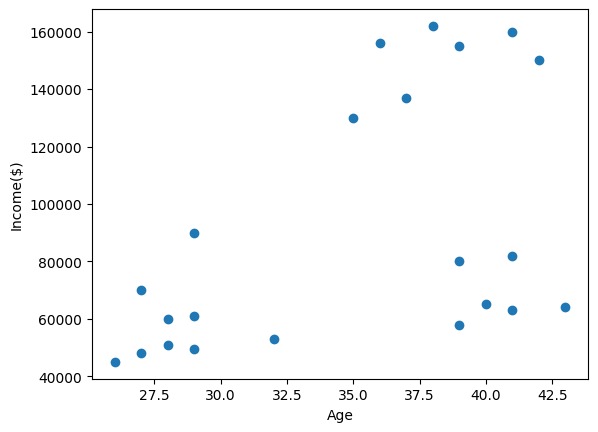
**KMeans Clustering**

import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
# kmeans clustering library  
from sklearn.cluster import KMeans  
from sklearn.preprocessing import StandardScaler

data = pd.read\_csv('income.csv')  
data

Name Age Income($)  
0 Rob 27 70000  
1 Michael 29 90000  
2 Mohan 29 61000  
3 Ismail 28 60000  
4 Kory 42 150000  
5 Gautam 39 155000  
6 David 41 160000  
7 Andrea 38 162000  
8 Brad 36 156000  
9 Angelina 35 130000  
10 Donald 37 137000  
11 Tom 26 45000  
12 Arnold 27 48000  
13 Jared 28 51000  
14 Stark 29 49500  
15 Ranbir 32 53000  
16 Dipika 40 65000  
17 Priyanka 41 63000  
18 Nick 43 64000  
19 Alia 39 80000  
20 Sid 41 82000  
21 Abdul 39 58000

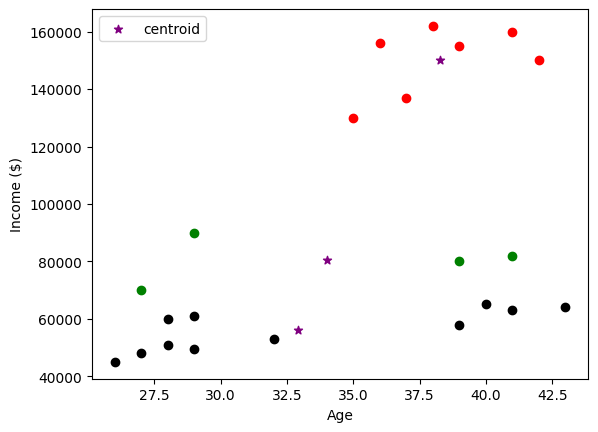
# plot the data  
plt.scatter(data['Age'], data['Income($)'])  
plt.xlabel('Age')  
plt.ylabel('Income($)')  
plt.show()



# Perfforming KMeans clustering  
km = KMeans(n\_clusters=3)  
y\_predicted = km.fit\_predict(data[['Age', 'Income($)']])  
y\_predicted

array([0, 0, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2])

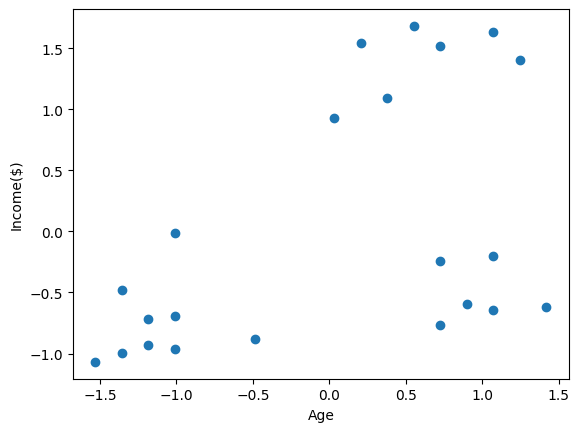
# create 3 dataframes for each cluster  
data['cluster'] = y\_predicted  
data1 = data[data.cluster==0]  
data2 = data[data.cluster==1]  
data3 = data[data.cluster==2]  
  
# plot the clusters  
plt.scatter(data1.Age, data1['Income($)'], color='green')  
plt.scatter(data2.Age, data2['Income($)'], color='red')  
plt.scatter(data3.Age, data3['Income($)'], color='black')  
plt.scatter(km.cluster\_centers\_[:,0], km.cluster\_centers\_[:,1], color='purple', marker='\*', label='centroid')  
plt.xlabel('Age')  
plt.ylabel('Income ($)')  
plt.legend()   
plt.show()



# Since the data is not scaled, the clustering is not accurate. We need to scale the data  
scaler = StandardScaler()  
scaler.fit(data[['Income($)']])  
data['Income($)'] = scaler.transform(data[['Income($)']])  
scaler.fit(data[['Age']])  
data['Age'] = scaler.transform(data[['Age']])  
data

Name Age Income($) cluster  
0 Rob -1.356055 -0.480684 0  
1 Michael -1.009157 -0.010159 0  
2 Mohan -1.009157 -0.692421 2  
3 Ismail -1.182606 -0.715947 2  
4 Kory 1.245679 1.401417 1  
5 Gautam 0.725332 1.519048 1  
6 David 1.072230 1.636679 1  
7 Andrea 0.551883 1.683732 1  
8 Brad 0.204985 1.542574 1  
9 Angelina 0.031536 0.930891 1  
10 Donald 0.378434 1.095575 1  
11 Tom -1.529504 -1.068841 2  
12 Arnold -1.356055 -0.998262 2  
13 Jared -1.182606 -0.927683 2  
14 Stark -1.009157 -0.962973 2  
15 Ranbir -0.488811 -0.880631 2  
16 Dipika 0.898781 -0.598316 2  
17 Priyanka 1.072230 -0.645368 2  
18 Nick 1.419128 -0.621842 2  
19 Alia 0.725332 -0.245422 0  
20 Sid 1.072230 -0.198369 0  
21 Abdul 0.725332 -0.762999 2

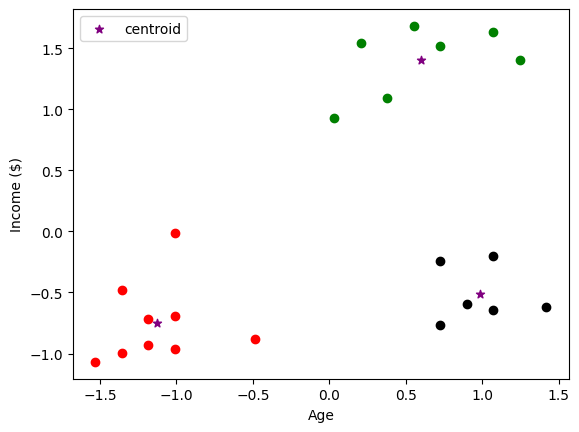
# plot the scaled data  
plt.scatter(data['Age'], data['Income($)'])  
plt.xlabel('Age')  
plt.ylabel('Income($)')  
plt.show()



# Perfforming KMeans clustering on scaled data  
km = KMeans(n\_clusters=3)  
y\_predicted = km.fit\_predict(data[['Age', 'Income($)']])  
y\_predicted

array([1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2])

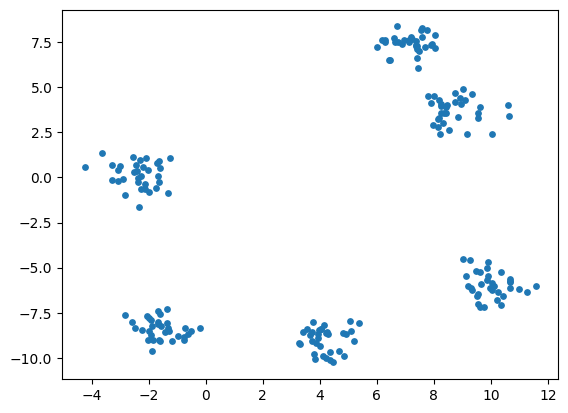
# plot the clusters  
data['cluster'] = y\_predicted  
data1 = data[data.cluster==0]  
data2 = data[data.cluster==1]  
data3 = data[data.cluster==2]  
plt.scatter(data1.Age, data1['Income($)'], color='green')  
plt.scatter(data2.Age, data2['Income($)'], color='red')  
plt.scatter(data3.Age, data3['Income($)'], color='black')  
plt.scatter(km.cluster\_centers\_[:,0], km.cluster\_centers\_[:,1], color='purple', marker='\*', label='centroid')  
plt.xlabel('Age')  
plt.ylabel('Income ($)')  
plt.legend()  
plt.show()



**DBScan Clustering**

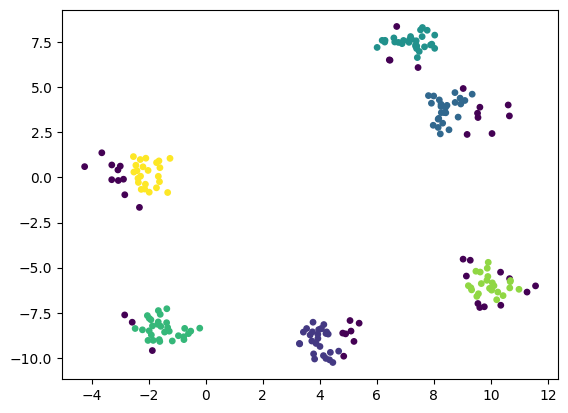
from sklearn.datasets import make\_blobs  
import matplotlib.pyplot as plt  
from sklearn.cluster import dbscan  
X, y\_true = make\_blobs(n\_samples=200, centers=6, cluster\_std=0.6)  
plt.scatter(X[:,0],X[:,1], s=15)

<matplotlib.collections.PathCollection at 0x21a7ad8e810>



dbs = dbscan(X)  
plt.scatter(X[:, 0], X[:, 1], c=dbs[1], s=15, cmap='viridis')

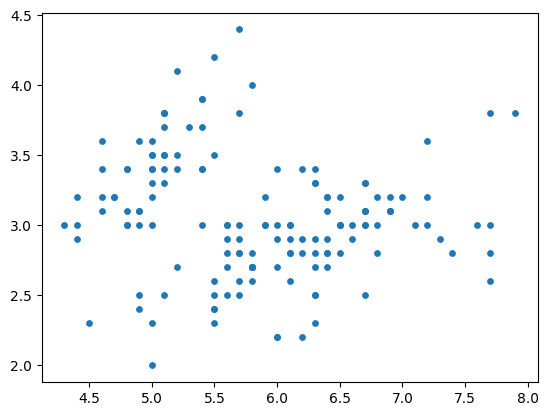
<matplotlib.collections.PathCollection at 0x21a799fc250>



# Perform DBScan on the iris dataset for 2 features  
from sklearn import datasets  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import DBSCAN  
iris = datasets.load\_iris()  
print(iris.feature\_names)  
  
# plot the iris dataset  
plt.scatter(iris.data[:,0], iris.data[:,1], s=15)

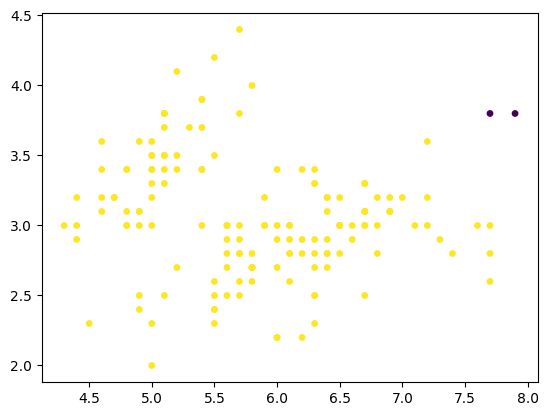
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

<matplotlib.collections.PathCollection at 0x21a7cf2c110>



# create a dataframe  
df = pd.DataFrame(iris.data, columns=iris.feature\_names)  
print(df.head())  
# Select the first 2 features  
X = df.iloc[:, [0, 1]].values  
# Perform DBScan  
dbs = DBSCAN(eps=0.5, min\_samples=5).fit(X)  
  
# Plot the clusters  
plt.scatter(X[:, 0], X[:, 1], c=dbs.labels\_, s=15, cmap='viridis')  
plt.show()

sepal length (cm) sepal width (cm) petal length (cm) petal width (cm)  
0 5.1 3.5 1.4 0.2  
1 4.9 3.0 1.4 0.2  
2 4.7 3.2 1.3 0.2  
3 4.6 3.1 1.5 0.2  
4 5.0 3.6 1.4 0.2



# EXPERIMENT-9

Aim: To performText Mining using textblob in python (TF-IDF generation, sentiment analysis, word-cloud,POStagging)

from textblob import TextBlob

from google.colab import drive  
drive.mount('/content/drive')

blob = TextBlob(" This could be a multiline text or paragraph copy from wiki or something else what ever you like. Good Bye")

blob.correct()

TextBlob(" His could be a multiline text or paragraph copy from wiki or something else what ever you like. Good Eye")

import nltk

nltk.download('punkt')  
nltk.download('wordnet')

[nltk\_data] Downloading package punkt to /root/nltk\_data...  
[nltk\_data] Unzipping tokenizers/punkt.zip.

True

nltk.download('averaged\_perceptron\_tagger')

[nltk\_data] Downloading package averaged\_perceptron\_tagger to  
[nltk\_data] /root/nltk\_data...  
[nltk\_data] Unzipping taggers/averaged\_perceptron\_tagger.zip.

True

nltk.download('brown')

[nltk\_data] Downloading package brown to /root/nltk\_data...  
[nltk\_data] Unzipping corpora/brown.zip.

True

blob.tags

[('This', 'DT'),  
 ('could', 'MD'),  
 ('be', 'VB'),  
 ('a', 'DT'),  
 ('multiline', 'JJ'),  
 ('text', 'NN'),  
 ('or', 'CC'),  
 ('paragraph', 'NN'),  
 ('copy', 'NN'),  
 ('from', 'IN'),  
 ('wiki', 'NN'),  
 ('or', 'CC'),  
 ('something', 'NN'),  
 ('else', 'RB'),  
 ('what', 'WP'),  
 ('ever', 'RB'),  
 ('you', 'PRP'),  
 ('like', 'VBP'),  
 ('Good', 'JJ'),  
 ('Bye', 'NNP')]

blob.noun\_phrases

WordList(['multiline text', 'paragraph copy', 'bye'])

blob.sentences

[Sentence(" This could be a multiline text or paragraph copy from wiki or something else what ever you like."),  
 Sentence("Good Bye")]

blob.words

WordList(['This', 'could', 'be', 'a', 'multiline', 'text', 'or', 'paragraph', 'copy', 'from', 'wiki', 'or', 'something', 'else', 'what', 'ever', 'you', 'like', 'Good', 'Bye'])

blob.word\_counts

defaultdict(int,  
 {'this': 1,  
 'could': 1,  
 'be': 1,  
 'a': 1,  
 'multiline': 1,  
 'text': 1,  
 'or': 2,  
 'paragraph': 1,  
 'copy': 1,  
 'from': 1,  
 'wiki': 1,  
 'something': 1,  
 'else': 1,  
 'what': 1,  
 'ever': 1,  
 'you': 1,  
 'like': 1,  
 'good': 1,  
 'bye': 1})

import nltk  
from textblob import TextBlob, Word, Sentence  
from nltk.corpus import wordnet as wn  
from textblob.wordnet import VERB,ADJ,ADV,NOUN  
for s in blob.sentences:  
 print(s.sentiment)  
 print(s.words[0].singularize())  
 print(s.words[-1].pluralize())

Sentiment(polarity=0.0, subjectivity=0.0)  
Thi  
likes  
Sentiment(polarity=0.7, subjectivity=0.6000000000000001)  
Good  
Byes

nltk.download('wordnet')

[nltk\_data] Downloading package wordnet to /root/nltk\_data...  
[nltk\_data] Package wordnet is already up-to-date!

True

w=Word("octopi")  
print(w.lemmatize())  
w=Word("done")  
print(Word("Beautiful").definitions)  
print(w)  
w=Word("Smart")  
print(w.get\_synsets(pos=wn.ADJ))

dreaming  
['delighting the senses or exciting intellectual or emotional admiration', '(of weather) highly enjoyable']  
done  
[Synset('smart.a.01'), Synset('chic.s.01'), Synset('bright.s.03'), Synset('fresh.s.12'), Synset('smart.s.05'), Synset('smart.s.06'), Synset('smart.s.07')]

---------------------------------------------------------------------------  
AttributeError Traceback (most recent call last)  
<ipython-input-39-bdb866cab31f> in <cell line: 1>()  
----> 1 w.get\_synset(pos=ADJ)  
  
AttributeError: 'Word' object has no attribute 'get\_synset'

print(w)

nltk.download('wordnet')

[nltk\_data] Downloading package wordnet to /root/nltk\_data...  
[nltk\_data] Package wordnet is already up-to-date!

True

File "<ipython-input-34-9d41c5fda68f>", line 2  
 s.words[-1].pluralize()  
 ^  
IndentationError: unexpected indent

# EXPERIMENT-10

Aim: To perform basic Web-Scraping tasks using the Beautiful soup package in Python