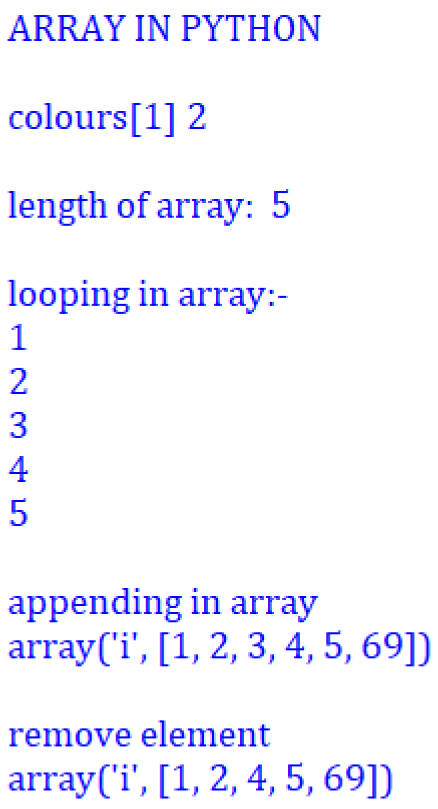
PROBLEM STATEMENT/AIM:

```
#python exercises
#1. array,list,set,dictionary
#2. modules and function
#3. file handling
#4. exception handling
#5. inheritance
```

SOURCE CODE:

```
import pickle as pk
import array as arr
import time
import math
import random

#1(a) array
print("ARRAY IN PYTHON")
colours = arr.array("i",[1,2,3,4,5])
x=colours[1]
print("colours[1]",x)
y=len(colours)
print("length of array: ",y)
print("looping in array:-")
for i in colours:
    print(i)
print("appending in array")
colours.append(69)
print(colours)
print("remove element")
colours.pop(2)
print(colours)
```

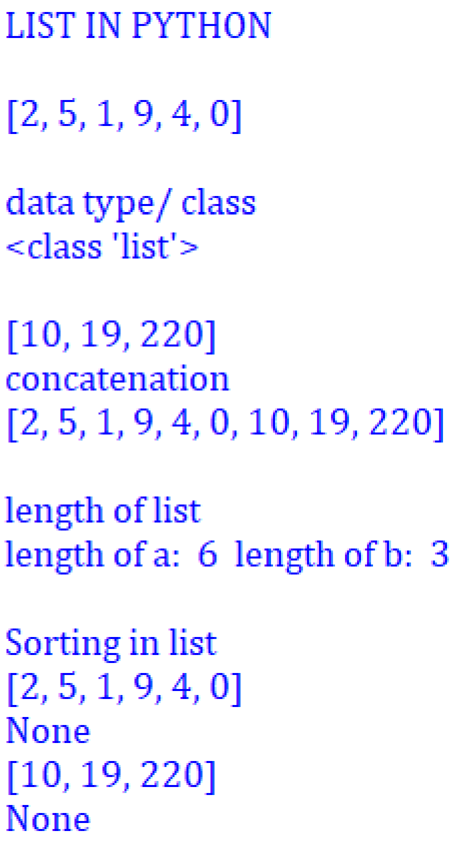
OUTPUT:



```
ARRAY IN PYTHON

colours[1] 2

length of array:  5

looping in array:-
1
2
3
4
5

appending in array
array('i', [1, 2, 3, 4, 5, 69])

remove element
array('i', [1, 2, 4, 5, 69])
```

```
#1(b) list
print("LIST IN PYTHON")
a=[2,5,1,9,4,0]
print(a)
a=[2,5,1,9,4,0,2,6,4,4]
print(a)
print("data type/ class")
print(type(a))
b=[10,19,220]
print(b)
print("concatenation")
print(a+b)
print("length of list")
print("length of a: ",len(a)," length of b: ",len(b))
print("Sorting in list")
print(a)
print(a.sort())
print(b)
print(b.sort())
```

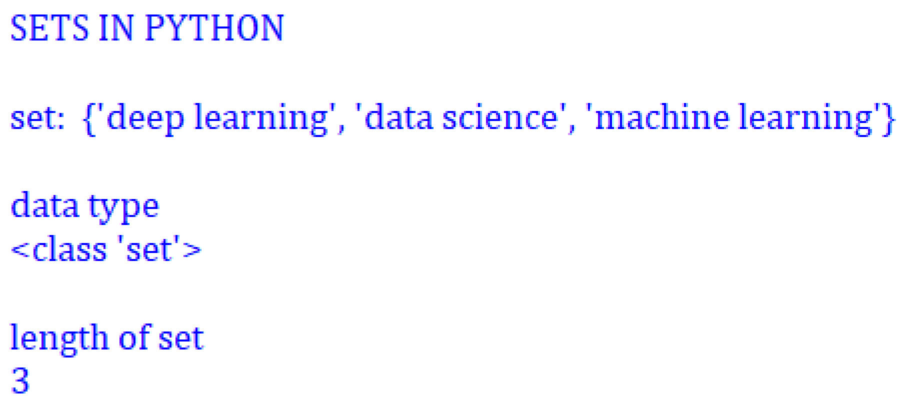
OUTPUT:



```
LIST IN PYTHON

[2, 5, 1, 9, 4, 0]

data type/ class
<class 'list'>

[10, 19, 220]
concatenation
[2, 5, 1, 9, 4, 0, 10, 19, 220]

length of list
length of a:  6  length of b:  3

Sorting in list
[2, 5, 1, 9, 4, 0]
None
[10, 19, 220]
None
```

```
#1(c) set
print("SETS IN PYTHON")
c={"data science","machine learning","deep learning"}
print("set: ",c)
print("data type")
print(type(c))
print("length of set")
print(len(c))
```

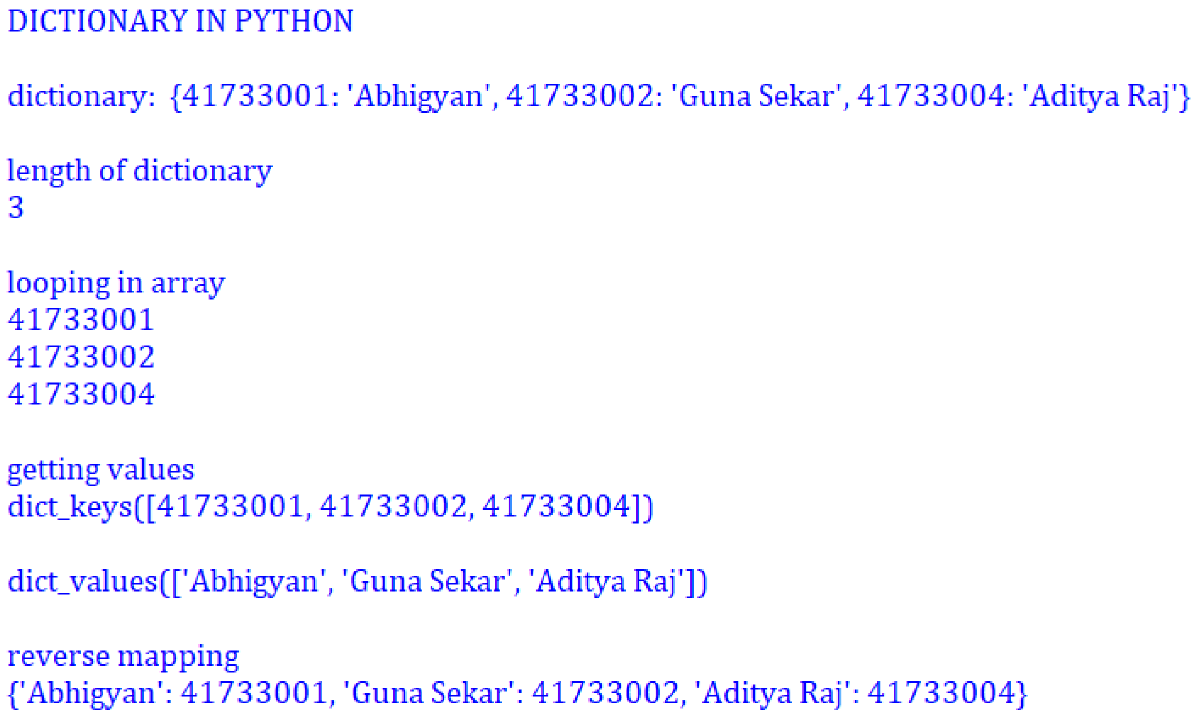
OUTPUT:



```
SETS IN PYTHON

set:  {'deep learning', 'data science', 'machine learning'}

data type
<class 'set'>

length of set
3
```

```
#1(d) dictionary
print("DICTIONARY IN PYTHON")
d={41733001:"Abhigyan",41733002:"Guna Sekar",41733004:"Aditya Raj"}
print("dictionary: ",d)
print("length of dictionary")
print(len(d))
print("looping in array")
for i in d:
    print(i)
print("getting values")
print(d.keys())
print(d.values())
print("reverse mapping")
e={v:k for k,v in d.items()}
print(e)
```

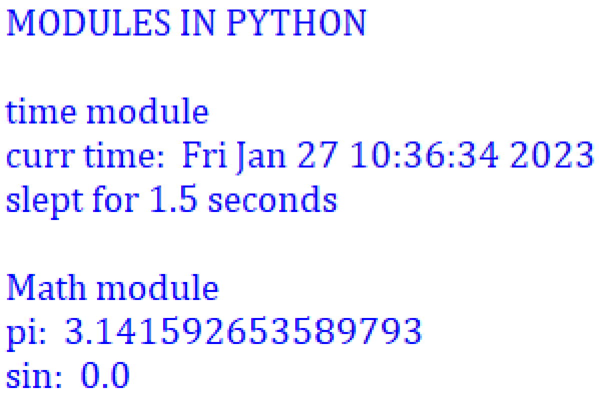
OUTPUT:



```
DICTIONARY IN PYTHON

dictionary:  {41733001: 'Abhigyan', 41733002: 'Guna Sekar', 41733004: 'Aditya Raj'}

length of dictionary
3

looping in array
41733001
41733002
41733004

getting values
dict_keys([41733001, 41733002, 41733004])

dict_values(['Abhigyan', 'Guna Sekar', 'Aditya Raj'])

reverse mapping
{'Abhigyan': 41733001, 'Guna Sekar': 41733002, 'Aditya Raj': 41733004}
```

#2(a) modules

```
print("MODULES IN PYTHON\n")
print("time module")
print("curr time: ",time.ctime(time.time()))
time.sleep(3)
print("slept for 3 seconds")
print("Math module")
print("pi: ",math.pi)
print("sin: ",math.sin(0))
```

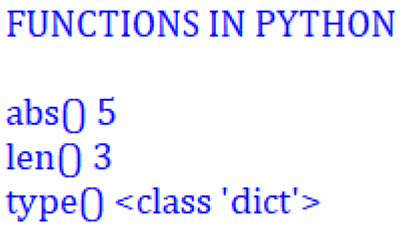


```
MODULES IN PYTHON

time module
curr time:  Fri Jan 27 10:36:34 2023
slept for 1.5 seconds

Math module
pi:  3.141592653589793
sin:  0.0
```

OUTPUT:

```
#2(b) functions
print("FUNCTIONS IN PYTHON")
print("abs()",abs(-5))
print("len()",len(d))
print("type()",type(d))
```

OUTPUT:



```
FUNCTIONS IN PYTHON

abs() 5
len() 3
type() <class 'dict'>
```

```
#3 File Handling


print("FILE HANDLING IN PYTHON")
"""Twinkle, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky."""


with open("poem.txt","r+") as file:
    print("readline(): ",file.readline())
    print("readlines(): ",file.readlines())
    print("write(): ",file.write("Sathyabama University"))
    print("writelines(): ",file.writelines(["BE CSE Data Science","BE CSE AI
ML","BE EEE"]))
    file.seek(0)
    print(file.readlines())
```

```
with open("binary.dat","wb+") as file:
    print("dump():  ",d)
    pk.dump(d,file)
    file.seek(0)
    print("load(): ",pk.load(file))
```

OUTPUT:

```
FILE HANDLING IN PYTHON

readline():  Twinkle, twinkle, little star,


readlines():  ['How I wonder what you are!\n', 'Up above the world so high,\n', 'Like a diamond in the sky.']

write():  22

writelines():  None

['Twinkle, twinkle, little star,\n', 'How I wonder what you are!\n', 'Up above the world so high,\n', 'Like a diamond in the sky.Sathyabama University  BE CSE Data Science BE CSE AI ML BE EE
E']
```

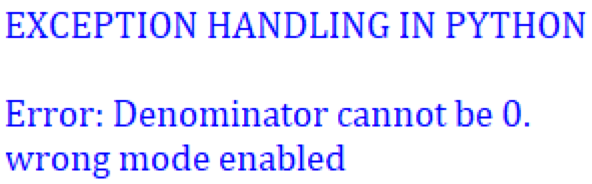
```
#4 Exception handling
#4(a)
try:
    numerator = 10
    denominator = 0
    result = numerator/denominator
    print(result)
except:
    print("Error: Denominator cannot be 0.")
```

# Output: Error: Denominator cannot be 0.

#4(b)

```
try:
    with open("binary1.dat","rb+") as file:
        print("dump():  ",d)
        pk.dump(d,file)
        print("executed successfully")
 except:
    print("wrong mode enabled")
```

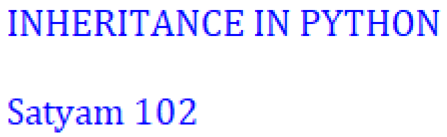
OUTPUT:



```
EXCEPTION HANDLING IN PYTHON

Error: Denominator cannot be 0.
wrong mode enabled
```

```
#5. inheritance
print("INHERITANCE IN PYTHON")
class Person(object):
    def __init__(self, name, id):
            self.name = name
            self.id = id
    def Display(self):
            print(self.name, self.id)
emp = Person("Satyam", 102)
emp.Display()
```

OUTPUT:



```
INHERITANCE IN PYTHON

Satyam 102
```

RESULT:

Thus the program was executed and output was verified successfully.

PROG 2(b)

AIM:

To read a CSV file and perform the following operations:

-display the description of the file

-display the file column wise

-display the first five entries of the file

-display the last five entries of the file

-display the size of particular column

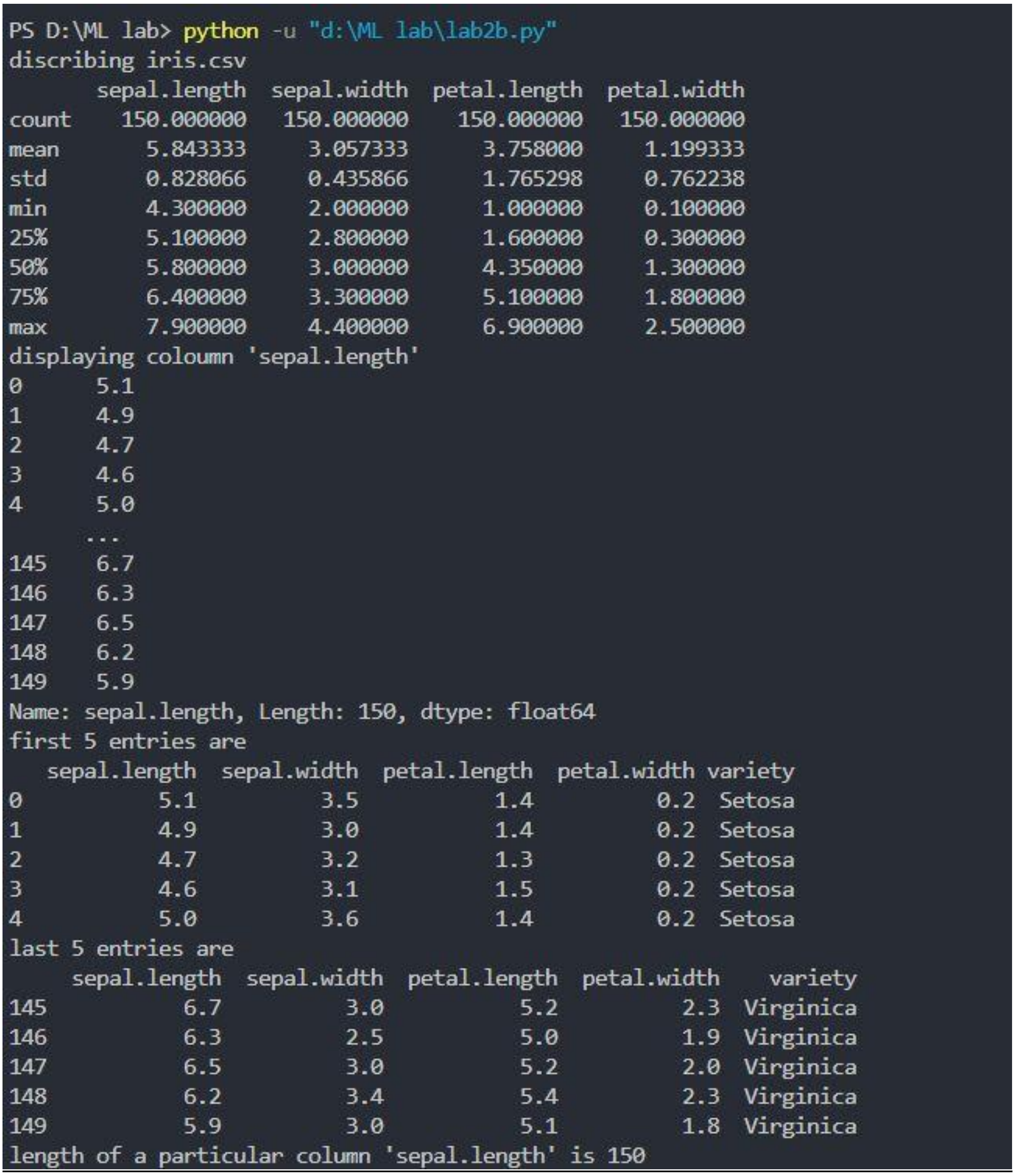
SOURCE:

Iris.csv: https://gist.githubusercontent.com/curran/a08a1080b88344b0c8a7/raw/0e7a9b0a5d22642a06d3d5b9bcbad9890c8ee534/iris.csv

SOURCE CODE:

```
import pandas as pd

df = pd.read_csv("./iris.csv")

print(f"discribing iris.csv\n{df.describe()}")

print(f"displaying coloumn 'sepal.length'\n{df['sepal.length']}")

print(f"first 5 entries are \n{df.head(5)}")

print(f"last 5 entries are \n{df.tail(5)}")

print(f"length of a particular column 'sepal.length' is len(df['sepal.length'])}")
```

OUTPUT:



```
PS D:\ML lab> python -u "d:\ML lab\lab2b.py"
discribing iris.csv
       sepal.length  sepal.width  petal.length  petal.width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
displaying coloumn 'sepal.length'
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: sepal.length, Length: 150, dtype: float64
first 5 entries are
   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
last 5 entries are
     sepal.length  sepal.width  petal.length  petal.width    variety
145           6.7          3.0           5.2          2.3  Virginica
146           6.3          2.5           5.0          1.9  Virginica
147           6.5          3.0           5.2          2.0  Virginica
148           6.2          3.4           5.4          2.3  Virginica
149           5.9          3.0           5.1          1.8  Virginica
length of a particular column 'sepal.length' is 150
```

RESULT:

Thus the program was executed and output was verified successfully.

PROG 2(c)

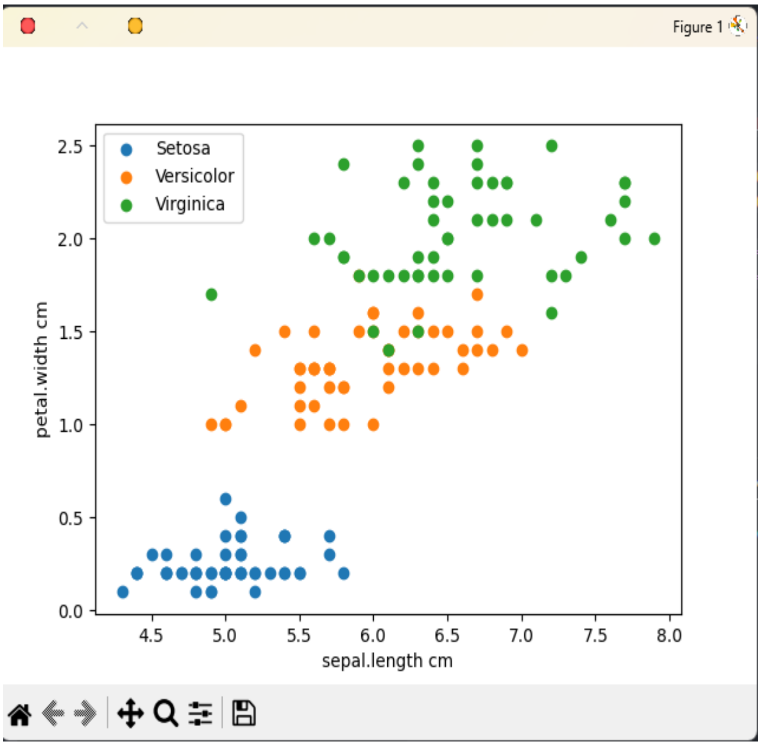
AIM:

Upload iris data set and create a scatter port using sepal length and petal width to separate the spices class

SOURCE CODE:

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("./iris.csv")
species=df["variety"].unique()
for i in species:
   x = df.loc[df["variety"]==i]["sepal.length"]
   y = df.loc[df["variety"]==i]["petal.width"]
   plt.scatter(x,y)
plt.legend(species)
plt.xlabel("sepal.length cm")
plt.ylabel("petal.width cm")

plt.show()
```

OUTPUT:



RESULT: Thus, the program is completed and the output is verified successfully.

PROG 3(a) CONFUSION MATRIX

AIM:

Evaluation matrix of ML algorithm using confusion matrix ; precision, accuracy, recall, specificity, sensitivity

SOURCE CODE:

```
import pandas as pd

import sklearn.linear_model as sk

import sklearn.model_selection as md

from sklearn import metrics


df = pd.read_csv("./iris.csv")

df = df.drop(df[df["variety"] == "Setosa"].index)

uni = df["variety"].unique()

df["variety"] = df["variety"].replace(uni, [0, 1])

x = df.iloc[:, :4]

y = df["variety"]


X_train, X_test, y_train, y_test = md.train_test_split(

   x, y, test_size=0.4,random_state=5)



logreg = sk.LogisticRegression()

logreg.fit(X_train, y_train)

y_pred = logreg.predict(x)


tm, fm = metrics.confusion_matrix(y, y_pred)

tn = tm[0]

fn = tm[1]

fp = fm[0]

tp = fm[1]
```

```
accuracy = (tp + tn)/(tp + tn + fp + fn)
sensitivity = tp/(tn + fn)
sensitivity = "{:2f}".format(sensitivity)
precision = tp/(tp + fp)
precision = "{:2f}".format(precision)
specificity = tn/(tn + fp)

print({"Accuracy": accuracy,
      "Precision": precision,
      "Sensitivity": sensitivity,
      "Specificity": specificity})
```

OUTPUT:

```
{'Accuracy': 0.97, 'Precision': '0.980000', 'Sensitivity': '0.980000', 'Specificity':
0.9795918367346939}
```

RESULT:
Thus the program was executed successfully and the output was verified.

PROG 3(b)

AIM:

To write a python function even digit(lower, upper)which will find all numbers between the lower and upper limit such that each digit of number is an even number.

SOURCE CODE:

```
L = []


def evendigit1(lower, upper):
    for i in range(lower, upper+1):
        L.append(i)
        for j in str(i):
            if int(j) % 2 != 0:
                L.remove(i)
                break
    return L



a, b = map(int, input("Enter Lower and Upper Limit: ").split())


y = evendigit1(a, b)
print(y)
```

OUTPUT:

```
Enter Lower and Upper Limit: 0 100
[0, 2, 4, 6, 8, 20, 22, 24, 26, 28, 40, 42, 44, 46, 48, 60, 62, 64, 66, 68, 80, 82, 84, 86, 88]
```

RESULT:

Thus the program is executed and the output is verified successfully.

PROG 4 KNN CLASSIFICATION

AIM:

To upload Iris.csv with three features sepal length, Sepal width and species, read the Test data from user to apply KNN Classification Algorithm and Predict the Test case (Assume K=5).without using predefined function.

SOURCE CODE:

```
import  pandas as pd

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error



data = pd.read_csv('./iris.csv')

train_data,test_data =train_test_split(data,test_size=0.2,random_state=5)

x_train,y_train = train_data.loc[:,["sepal.length","petal.width"]],train_data['variety']

x_test,y_test = test_data.loc[:,["sepal.length","petal.width"]],test_data['variety']



model  = KNeighborsClassifier(n_neighbors=5)

model.fit(x_train,y_train)

predicted_data = model.predict(x_test)


def replacing_catagory(l):

  for i,e in enumerate(l) :

    if e == 'Versicolor':

      l[i]=1

    if e == 'Setosa':

      l[i]=0

    if e == 'Virginica':

      l[i]=2

  return l
```

```
y_test = replacing_catagory(np.array(y_test))
predicted_data = replacing_catagory(predicted_data)
mse = mean_squared_error(y_test, predicted_data)



print('The mean squared error is ',mse)
```

OUTPUT:

```
The mean squared error is  0.1
```

RESULT:

Thus the program was executed and the output was verified successfully.

PROG 5 NAIVE BAYES CLASSIFICATION

AIM:

To upload Iris.csv dataset with sepal length, Sepal width, petal length, petal width.

Employee Naive Bayes Classification methodology and predict the type of species for a given set of attributes.

Note : Do not use Predefined Function

SOURCE CODE:

```
import pandas as pd

import numpy as np

from sklearn.metrics import confusion_matrix, f1_score

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error



data = pd.read_csv("./iris.csv")

data = data.drop(data[data["variety"] == "Setosa"].index)

uni = data["variety"].unique()

data["variety"] = data["variety"].replace(uni, [0, 1])



def calculate_prior(df, Y):

    classes = sorted(list(df[Y].unique()))

    prior = []

    z = len(df)

    for i in classes:

        prior.append(len(df[df[Y] == i])/z)

    return prior



def calculate_likelihood_gaussian(df, feat_name, feat_val, Y, label):

    feat = list(df.columns)
    df = df[df[Y]==label]
    mean, std = df[feat_name].mean(), df[feat_name].std()
    p_x_given_y = (1 / (np.sqrt(2 * np.pi) * std)) *  np.exp(-((feat_val-mean)*2 / (2 * std*2 )))
    return p_x_given_y


def naive_bayes_gaussian(df, X, Y):
    # get feature names
    features = list(df.columns)[:-1]


    # calculate prior
    prior = calculate_prior(df, Y)


    Y_pred = []
    # loop over every data sample
    for x in X:
        # calculate likelihood
        labels = sorted(list(df[Y].unique()))
        likelihood = [1]*len(labels)
        for j in range(len(labels)):
            for i in range(len(features)):
                likelihood[j] *= calculate_likelihood_gaussian(df, features[i], x[i], Y, labels[j])


        # calculate posterior probability (numerator only)
        post_prob = [1]*len(labels)
        for j in range(len(labels)):
            post_prob[j] = likelihood[j] * prior[j]


        Y_pred.append(np.argmax(post_prob))


    return np.array(Y_pred)
```

```
train, test = train_test_split(data, test_size=.2, random_state=41)


X_test = test.iloc[:,:-1].values
Y_test = test.iloc[:,-1].values
Y_pred = naive_bayes_gaussian(train, X=X_test, Y="variety")



print("The accuracy of the model is",f1_score(Y_test, Y_pred))
```

OUTPUT:

```
The accuracy of the model is 0.8181818181818181
```

RESULT:

Thus the program is executed and the output is verified successfully.

Decision tree classification methodologies

Using iris dataset

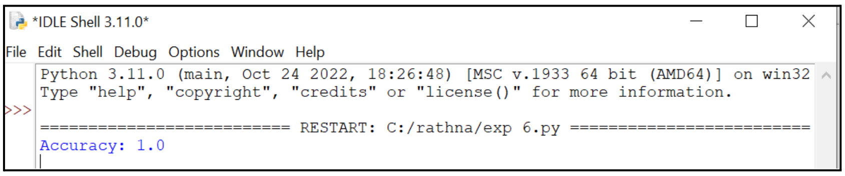
Prog - 6

Aim: Employing Decision tree Classification methodologies to the iris dataset and plotting the result.

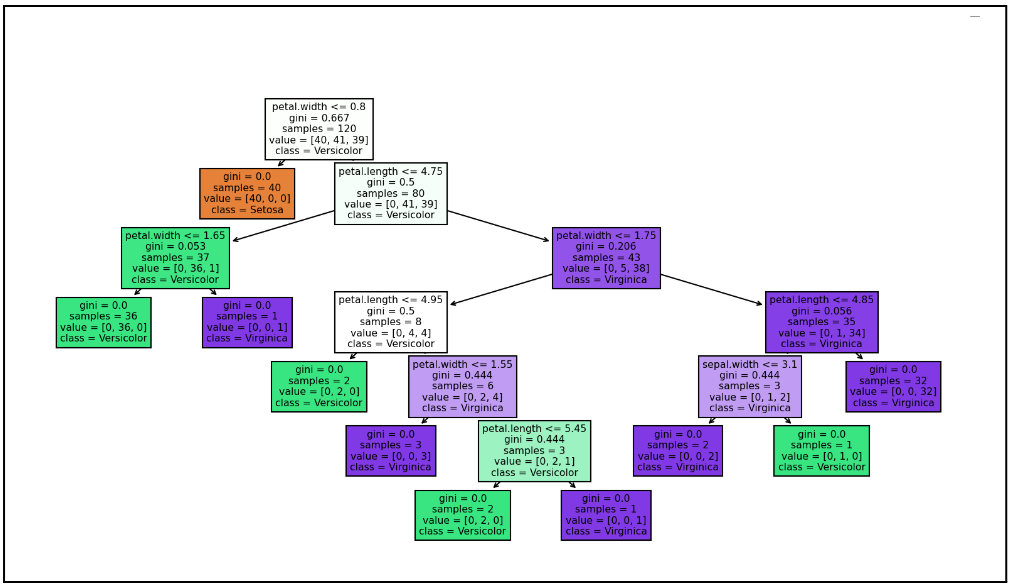
Source code:

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt

df = pd.read_csv('C:/iris.csv')

x = df.drop('variety', axis=1)
y = df['variety']

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=42)
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
y_pred = dt.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:",accuracy)

fig, ax = plt.subplots(figsize=(10,10))

tree.plot_tree(dt,feature_names=x.columns,class_names = np.unique(y),filled =
True, ax = ax)

plt.show()
```

Output:



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========================== RESTART: C:/rathna/exp 6.py =========================
Accuracy: 1.0
```



PROG 7 ANALYSIS AND INTERPRETATION OF DATA

AIM

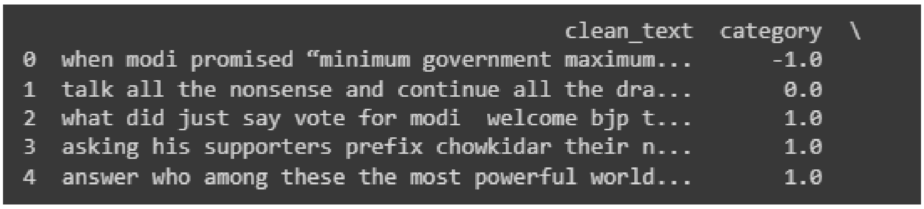
To manipulate the Twitter Data Set by removing the Punctuation, Numbers, Special Characters and word length<=3, tokenize the Words and Stem and generate a word cloud for the Twitter dataset and retrieve the top 15 positive and negative tags.

SOURCE CODE

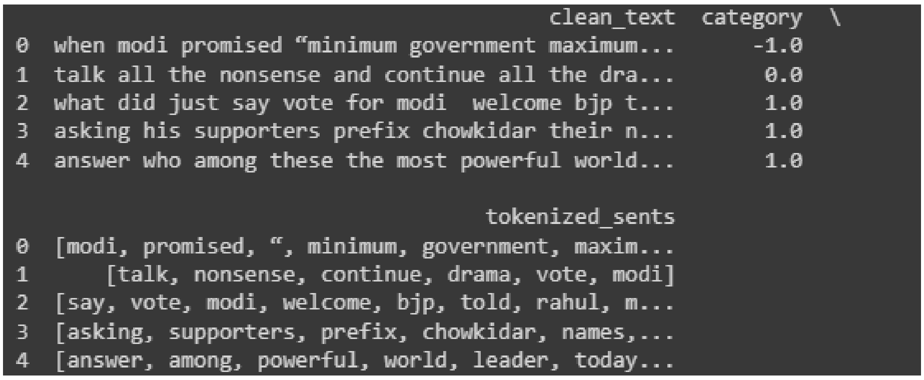
```
import numpy as np

import pandas as pd

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

import nltk

from wordcloud import WordCloud, STOPWORDS

import matplotlib.pyplot as plt

data=pd.read_csv('Twitter_Data.csv')

data=data.iloc[:50,:]

nltk.download('punkt')

nltk.download('stopwords')


print(data.head())


stop_words = set(stopwords.words('english'))


data['tokenized_sents'] = data.apply(lambda row:
nltk.word_tokenize(str(row['clean_text'])), axis=1)

for i in range(50):

   words = data.loc[i,'tokenized_sents']

   wordsFiltered = []

   for w in words:

      if w not in stop_words:

         wordsFiltered.append(w)
```

```
    data.at[i,'tokenized_sents']=wordsFiltered


print(data.head())


comment_words=''
for i in range(50):
    tokens=data.iloc[i,2]
    comment_words += " ".join(tokens)+" "


wordcloud = WordCloud(collocations = False, background_color =
'white').generate(comment_words)


# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)


plt.show()
```

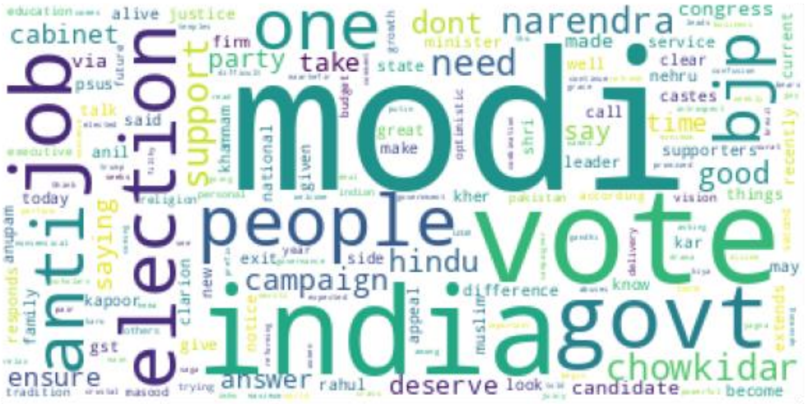
OUTPUT



```
                                          clean_text  category  \
0  when modi promised “minimum government maximum...      -1.0
1  talk all the nonsense and continue all the dra...       0.0
2  what did just say vote for modi  welcome bjp t...       1.0
3  asking his supporters prefix chowkidar their n...       1.0
4  answer who among these the most powerful world...       1.0
```



```
                                          clean_text  category  \
0  when modi promised “minimum government maximum...      -1.0
1  talk all the nonsense and continue all the dra...       0.0
2  what did just say vote for modi  welcome bjp t...       1.0
3  asking his supporters prefix chowkidar their n...       1.0
4  answer who among these the most powerful world...       1.0

                                     tokenized_sents
0  [modi, promised, “, minimum, government, maxim...
1      [talk, nonsense, continue, drama, vote, modi]
2  [say, vote, modi, welcome, bjp, told, rahul, m...
3  [asking, supporters, prefix, chowkidar, names,...
4  [answer, among, powerful, world, leader, today...
```



RESULT:

Thus the program is executed and output is verified successfully.

PROG 8 LINEAR REGRESSION

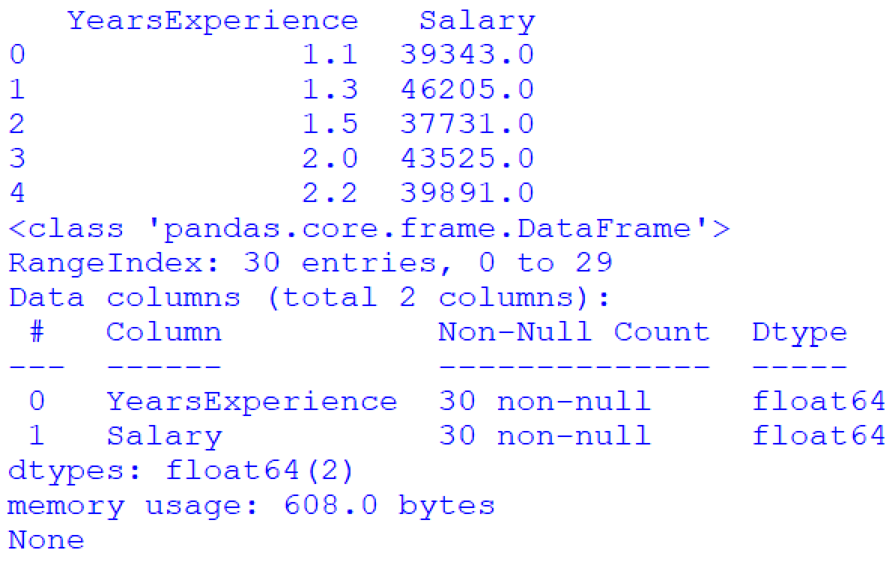
AIM:

To write a python program to compute linear regression curve between salary and experience from a csv dataset.

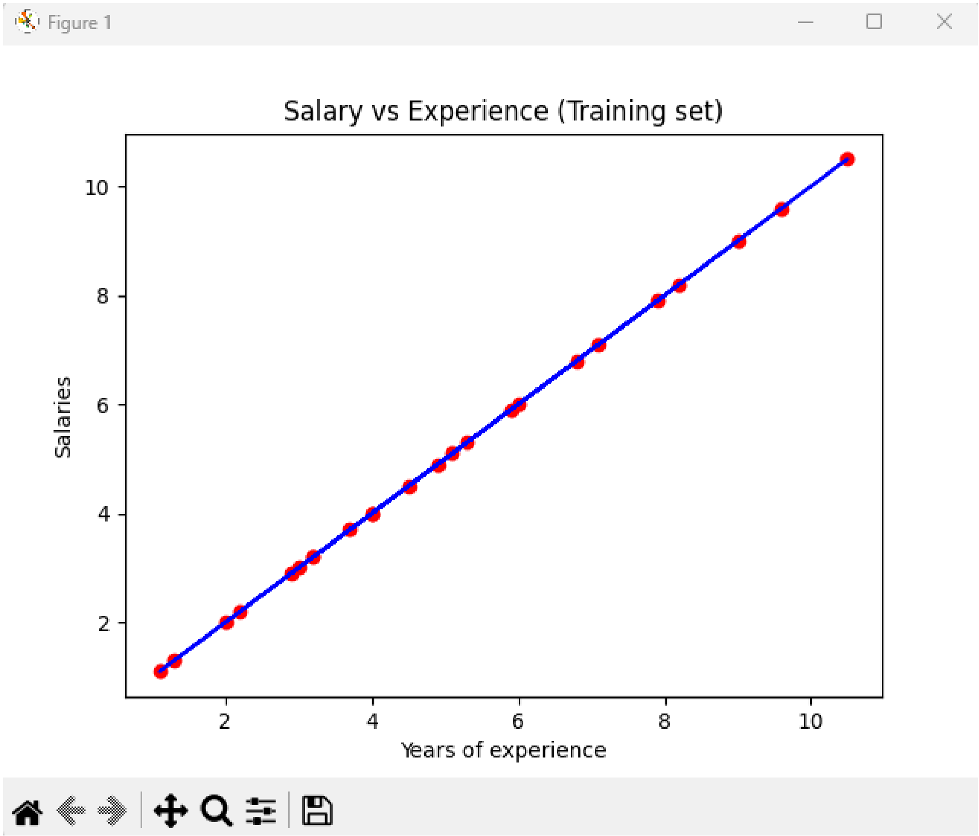
SOURCE CODE:

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

dataset = pd.read_csv("./Salary_Data.csv")
print(dataset.head())
print(dataset.info())

X = dataset.iloc[:,:-1].values  #independent variable array
y = dataset.iloc[:,:-1].values  #dependent variable vector

# splitting the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=0)
#print('Training Data\n',X_train)
#print('Testing Data\n',X_test)

# fitting the regression model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train) #actually produces the linear eqn for the data

# Plotting the graph for the Training dataset

plt.scatter(X_train,y_train,color='red') # plotting the observation line
plt.plot(X_train, regressor.predict(X_train), color='blue') # plotting the regression line

plt.title("Salary vs Experience (Training set)") # stating the title of the graph


plt.xlabel("Years of experience") # adding the name of x-axis

plt.ylabel("Salaries") # adding the name of y-axis

plt.show() # specifies end of graph


# Plotting the graph for the Testing dataset


plt.scatter(X_test, y_test, color='red')

plt.plot(X_train, regressor.predict(X_train), color='blue') # plotting the regression line

plt.title("Salary vs Experience (Testing set)")


plt.xlabel("Years of experience")

plt.ylabel("Salaries")

plt.show()

plt.show()
```

OUTPUT:



```
   YearsExperience   Salary
0              1.1  39343.0
1              1.3  46205.0
2              1.5  37731.0
3              2.0  43525.0
4              2.2  39891.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
None
```



RESULT:

Thus the program is executed and the output is verified successfully.