Problem Statement 1.Find the Standard Deviation and Variance of the grouped data.

```
import math
low_lim=[0,10,20,30,40,50]   # lower limit
up_lim=[10,20,30,40,50,60]   # upper limit
freq=[27,10,7,5,4,2]         # m_ean=stat.mean(lower_values)
n=len(low_lim)               # length of the data
freq_sum=0
for i in range(0,n):
    freq_sum=freq_sum+freq[i]
xifi = 0
xifi2 = 0
for i in range(0,len(low_lim)):
    xi = (up_lim[i] + low_lim[i])/2
    xifi += (freq[i] * xi)
    xifi2 += (freq[i] * xi * xi)
xmean = xifi / freq_sum
ss = xifi2 - ((xifi ** 2)/freq_sum)
var = ss/(freq_sum-1)
sd = math.sqrt(var)
print("Variance of above data is:- ",var)
print("Standard Deviation of above data is:- ",sd)
```

Output:-

```
PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\SD.py"
Variance of above data is:-  222.55892255892255
Standard Deviation of above data is:-  14.918408848095112
```

Problem Statement 2.Implement Finds Algorithm using python

```
import csv
a =[]
with open("./data.csv","r") as csvfile:
    for row in csv.reader(csvfile):
        a.append(row)

    print("\nTraining instances:- ",len(a)-1)
    num_attr = len(a[0])-1
    print("\nTotal attributes:-",num_attr)
    print("\nInitial  Hypothesis:- ")
    hypothesis = ['0'] * num_attr
    print(hypothesis)
    for i in range(0, len(a)):
        if a[i][num_attr] == 'yes':
            for j in range(0, num_attr):
                if hypothesis[j] == '0' or hypothesis[j] == a[i][j]:
                    hypothesis[j] = a[i][j]
                else:
                    hypothesis[j] = "?"
                    print(hypothesis)
    print("\nThe hypothesis of the training instance {} is: \n".format(i),hypothesis)
    print("\nThe specific hypothesis for the training instance is: ")
    print(hypothesis)
```

Output:-

```
PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\Finds.py"
Training instances:-  4
Total attributes:- 6
Initial Hypothesis:-
['0', '0', '0', '0', '0', '0']
['sunny', 'warm', '?', 'strong', 'warm', 'same']
['sunny', 'warm', '?', 'strong', 'warm', 'same']
['sunny', 'warm', '?', 'strong', '?', 'same']
['sunny', 'warm', '?', 'strong', '?', '?']
The hypothesis of the training instance 4 is:
 ['sunny', 'warm', '?', 'strong', '?', '?']
The specific hypothesis for the training instance is:
['sunny', 'warm', '?', 'strong', '?', '?']
```

Problem Statement 3.Implement Candidate Elimination Algorithm using python.

```
import csv
with open("./data.csv","r") as csvfile:
    csvf = csv.reader(csvfile)
    data = list(csvf)


    s = data[1][:-1]
    print(s)
    g = [['?' for i in range(len(s))]for j in range(len(s))]
    for i in data:
        if i[-1] == "yes":
            for j in range(len(s)):
                if i[j] != s[j]:
                    s[j] = "?"
                    g[j][j] = "?"
        elif i[-1] == "no":
            for j in range(len(s)):
                if i[j] != s[j]:
                    g[j][j] = s[j]
                else:
                    g[j][j] = "?"
    gh = []
    for i in g:
        for j in i:
            if j != "?":
                gh.append(i)
    print("\n Specific hypothesis:- \n",s)
    print("\n General hypothesis:- \n",gh)
```

Output:-

```
PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\CE.py"
['sunny', 'warm', 'normal', 'strong', 'warm', 'same']
 Specific hypothesis:-
 ['sunny', 'warm', '?', 'strong', '?', '?']
 General hypothesis:-
 [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

Problem Statement 4.Implement Linear Regression using python and numpy library.

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

x = np.array([1,2,3,4,5])
y = np.array([7,14,15,18,19])
n = np.size(x)
x_mean = np.mean(x)
y_mean = np.mean(y)

Sxy = np.sum(x*y) - n*x_mean*y_mean
Sxx = np.sum(x*x) - n*x_mean*x_mean

b1 = Sxy/Sxx
b0 = y_mean - b1 * x_mean
print("\nSlope b1 is:- ",b1)
print("\nIntercept b0 is:- ",b0)

plt.scatter(x,y)
plt.xlabel("Independent variable X")
plt.ylabel("Dependent variable Y")

y_pred = b1 * x +b0
plt.scatter(x,y,color="red")
plt.plot(x,y_pred,color = "green")
plt.xlabel("X")
plt.ylabel("Y")

error = y - y_pred
se = np.sum(error ** 2)
print("\nSquared Error is ",se)

mse = se/n
print("\nMean Squared Error is ",mse)
rmse = np.sqrt(mse)
print("\nRoot mean Squared Error is ",rmse)
sst = np.sum((y-y_mean) ** 2)
r2 = 1 - (se/sst)
print("\nR square",r2)
plt.show()
```

Output:-

PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\LinearReg.py"

Slope b1 is:- 2.8

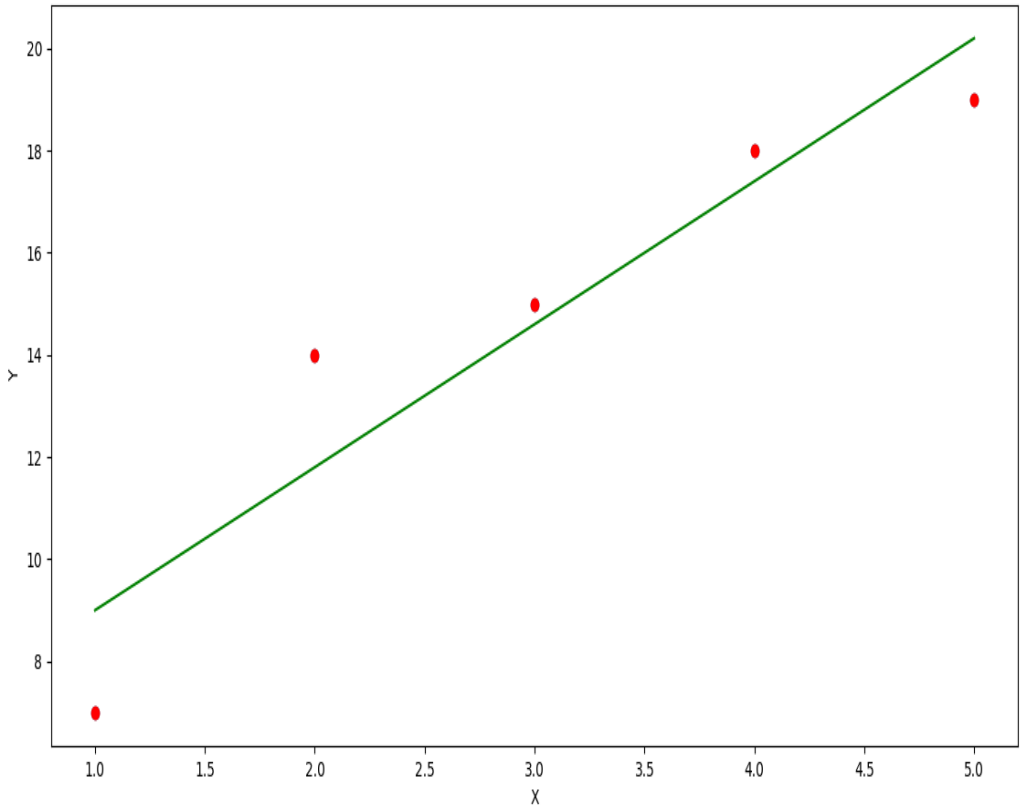
Intercept b0 is:- 6.200000000000001

Squared Error is 10.800000000000004

Mean Squared Error is 2.160000000000001

Root mean Squared Error is 1.4696938456699071

R square 0.8789237668161435



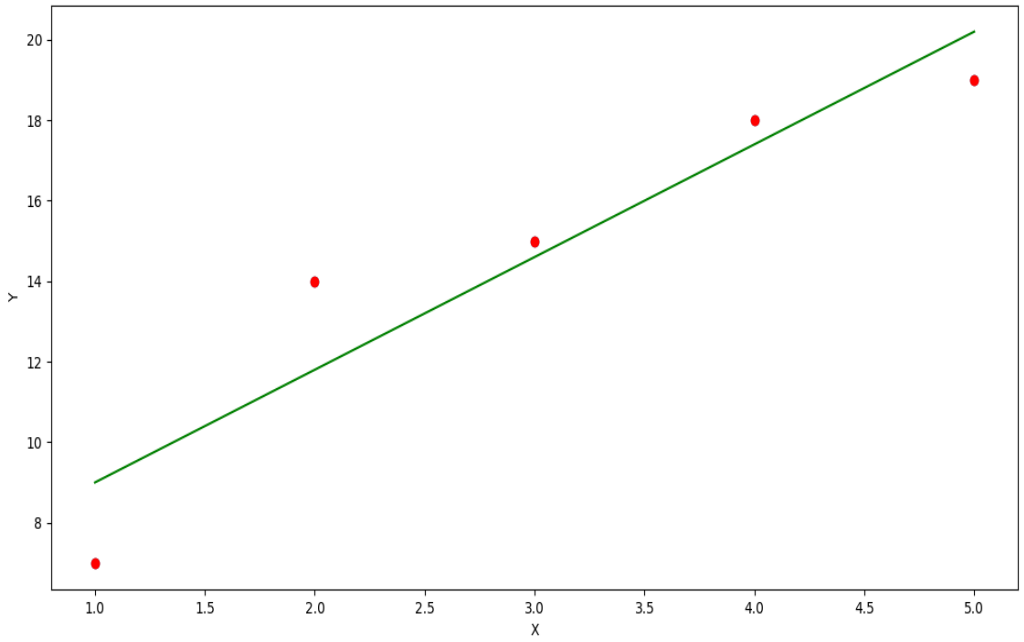
Problem Statement 5.Implement Linear Regression using python without using any library.

```
import matplotlib.pyplot as plt
import math
def mean(data):
    sum = 0
    for X in data:
        sum += X
    return sum/len(data)

x = [1,2,3,4,5]
y = [7,14,15,18,19]
n = len(x)
xm = mean(x)
ym = mean(y)
Sxy = 0
Sxx = 0
for i in range(len(x)):
    Sxy += (x[i] * y[i])
    Sxx +=(x[i] * x[i])
Sxy = Sxy - n * xm * ym
Sxx = Sxx - n * xm * xm
b1 = Sxy / Sxx
b0 = ym - b1 * xm
print("\nSlope b1 is:- ",b1)
print("\nIntercept b0 is:- ",b0)
yp = []
for i in range(len(x)):
    yp.append(b0 + (b1 * x[i]))
plt.scatter(x,y,color="black")
plt.plot(x,yp,color="green")
plt.xlabel("X")
plt.ylabel("Y")
err = []
for i in range(len(y)):
    err.append(y[i] - yp[i])
se = 0
for i in range(len(err)):
    se += (err[i] ** 2)
print("\n Squared Error is ",se)
```

```
mse = se/n
print("\nMean Squared Error is ",mse)
rmse= math.sqrt(mse)
print("\nRoot mean Squared Error is ",rmse)
sst = 0
for i in range(len(y)):
    sst += (y[i] - ym) ** 2
r2 = 1 - (se/sst)
print("\nR square",r2)
plt.show()
```

Output:-

```
PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\LR.py"
Slope b1 is:-  2.8
Intercept b0 is:-  6.200000000000001
Squared Error is  10.800000000000004
Mean Squared Error is  2.160000000000001
Root mean Squared Error is  1.4696938456699071
R square 0.8789237668161435
```

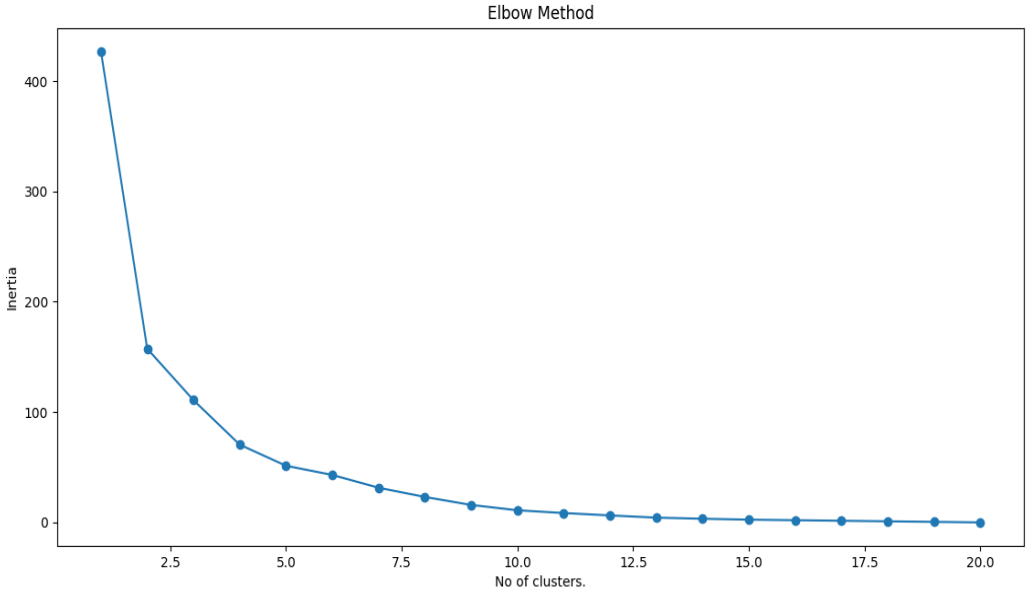


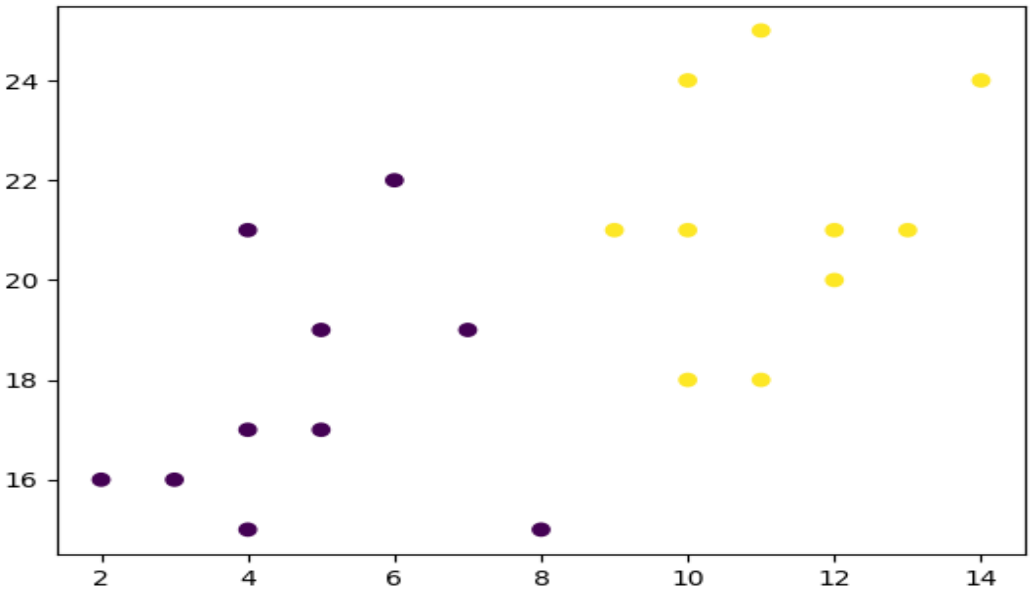
Problem Statement 6.Implement KMeans algorithm using python.

```
import matplotlib.pyplot as plt
import KMeans
from sklearn.cluster import KMeans

X = [4,5,10,4,3,11,14,6,10,12,2,4,5,10,12,13,9,8,7,11]
Y = [21,19,24,17,16,25,24,22,21,21,16,15,17,18,20,21,21,15,19,18]

data = list(zip(X,Y))
print(data)

inertias = []
for i in range(1,21):
    Kmeans = KMeans(n_clusters=i)
    Kmeans.fit(data)
    inertias.append(Kmeans.inertia_)
plt.plot(range(1,21),inertias, marker='o')
plt.title("Elbow Method")
plt.xlabel("No of clusters.")
plt.ylabel("Inertia")
plt.show()
Kmeans = KMeans(n_clusters=2)
Kmeans.fit(data)
plt.scatter(X,Y,c=Kmeans.labels_)
plt.show()
```

Output:-

PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\KMeans.py"
[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (6, 22), (10, 21), (12, 21), (2, 16), (4, 15), (5, 17), (10, 18), (12, 20), (13, 21), (9, 21), (8, 15), (7, 19), (11, 18)]



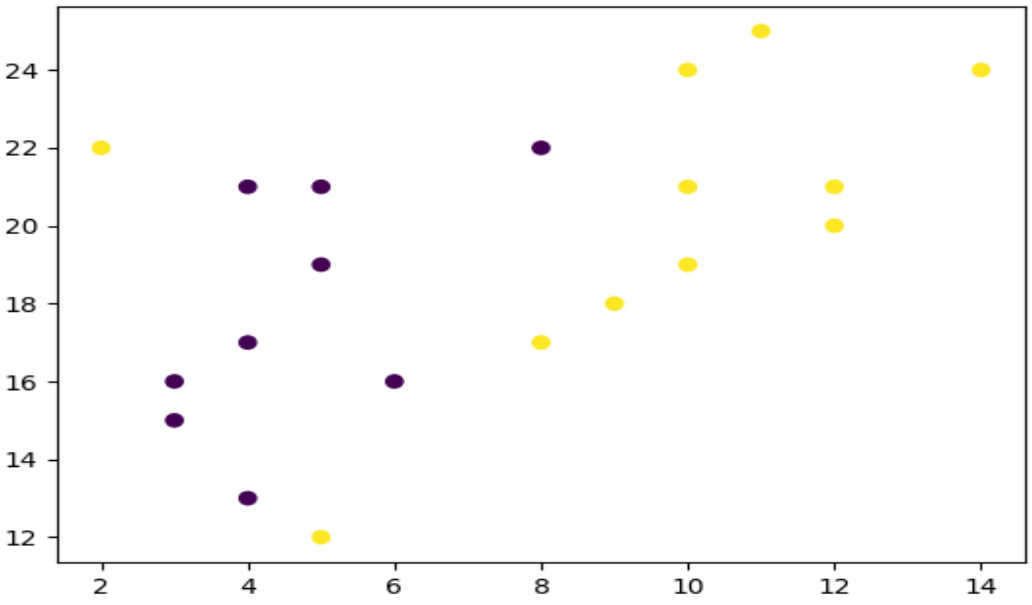


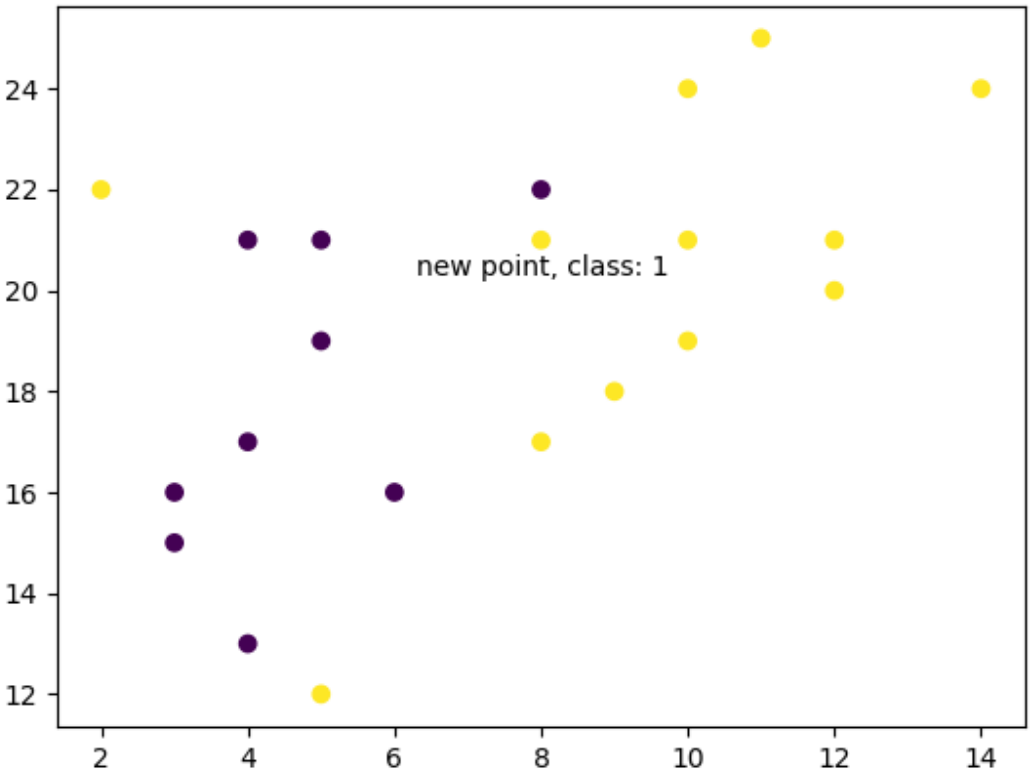
Problem Statement 7.Implement KNN algorithm using python.

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12, 6, 9, 12, 5, 2, 3, 10, 8, 4, 5]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21, 16, 18, 20, 21, 22, 15, 19, 17, 13, 12]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1]
data = list(zip(x, y))
print(data)
plt.scatter(x, y, c=classes)
plt.show()
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(data, classes)
new_x = 8
new_y = 21
new_point = [(new_x, new_y)]
prediction = knn.predict(new_point)
print(prediction)
plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()
```

Output:-

```
PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\KNN.py"
[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (8, 22), (10, 21), (12, 21), (6, 16), (9,
18), (12, 20), (5, 21), (2, 22), (3, 15), (10, 19), (8, 17), (4, 13), (5, 12)]
[1]
```


24
22
20
18
16
14
12
2
4
6
8
10
12
14


new point, class: 1
24
22
20
18
16
14
12
2
4
6
8
10
12
14

Problem Statement 8.ID3 algorithm to construct decision tree using python

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
data = pd.read_csv("./dataset.csv")
print("The first 5 value of data is\n",data.head())

x =data.iloc[:,:-1]
print("The first 5 value of train data is\n",x.head())
y=data.iloc[:,-1]
print("The first 5 value of train output is\n",y.head())
le_outlook = LabelEncoder()
x.outlook = le_outlook.fit_transform(x.outlook)
le_temp = LabelEncoder()
x.temp = le_temp.fit_transform(x.temp)
le_humidity = LabelEncoder()
x.humidity = le_humidity.fit_transform(x.humidity)
le_windy = LabelEncoder()
x.windy = le_windy.fit_transform(x.windy)
print("Now the train date is\n",x.head())
le_playtenis = LabelEncoder()
y = le_playtenis.fit_transform(y)
print("Now the train data (target variable) is\n",y)

X_train,X_test,ytrain,ytest = train_test_split(x,y,test_size=0.2,shuffle=False)
print("Features in training set is \n",X_train)
print("Test set is \n",X_test)
classifier = DecisionTreeClassifier(criterion='entropy')
classifier.fit(X_train,ytrain)
pred1 = classifier.predict(X_test)
print("For input \n {0},\n we obtain
{1}".format((X_test),le_playtenis.inverse_transform(pred1)))
print("Accuracy score is:- ",metrics.accuracy_score(ytest,pred1))
```

Output:-

```
PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\ID3.py"
The first 5 value of data is
     outlook  temp humidity  windy playtenis
0     sunny   hot     high  False        no
1     sunny   hot     high   True        no
2  overcast   hot     high  False       yes
3     rainy  mild     high  False       yes
4     rainy  cool   normal  False       yes
The first 5 value of train data is
     outlook  temp humidity  windy
0     sunny   hot     high  False
1     sunny   hot     high   True
2  overcast   hot     high  False
3     rainy  mild     high  False
4     rainy  cool   normal  False
The first 5 value of train output is
 0     no
1     no
2    yes
3    yes
4    yes
Name: playtenis, dtype: object
Now the train date is
    outlook  temp  humidity  windy
0        2     1         0      0
1        2     1         0      1
2        0     1         0      0
3        1     2         0      0
4        1     0         1      0
Now the train data (target vaiable) is
 [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
Features in training set is
     outlook  temp  humidity  windy
0        2     1         0      0
1        2     1         0      1
2        0     1         0      0
3        1     2         0      0
4        1     0         1      0
```

```
5        1    0        1     1
6        0    0        1     1
7        2    2        0     0
8        2    0        1     0
9        1    2        1     0
10       2    2        1     1
Test set is
    outlook  temp  humidity  windy
11       0    2        0     1
12       0    1        1     0
13       1    2        0     1
For input
    outlook  temp  humidity  windy
11       0    2        0     1
12       0    1        1     0
13       1    2        0     1,
 we obtain ['yes' 'yes' 'no']
Accuracy score is:-  1.0
```

Problem Statement 9. Logistic Regression Algorithm using python

```
import numpy as np
from sklearn import linear_model

x =
np.array([3.78,2.44,2.09,0.14,1.72,1.67,4.92,4.37,4.96,4.52,3.69,5.88,2.98,3.33]).reshape(-1,1)
y = np.array([0,0,0,0,0,0,1,1,1,1,1,1,0,0])
logr = linear_model.LogisticRegression()
logr.fit(x,y)
predicted = logr.predict(np.array([3.46]).reshape(-1,1))

print(predicted)

def logit2prob(logr,x):
    log_odds = logr.coef_*x+logr.intercept_
    odds = np.exp(log_odds)
    probability = odds/(1+odds)
    return probability

print(logit2prob(logr,x))
```

Output:-

```
PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\LogReg.py"
[0]
[[0.50904473]
 [0.11909549]
 [0.07356805]
 [0.00407939]
 [0.04328738]
 [0.04024636]
 [0.85437417]
 [0.71771631]
 [0.8617785 ]
 [0.76155203]
 [0.47486011]
 [0.96190453]
 [0.23504144]
 [0.34345083]]
```

Problem Statement 10. Implement Naïve Bayes Classifier using python.

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB

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

print("The first 5 values of data is :\n",data.head())
X = data.iloc[:,:-1]

print("\nThe First 5 values of train data is\n",X.head())
y = data.iloc[:,-1]

print("\nThe first 5 values of Train output is\n",y.head())

le_outlook = LabelEncoder()
X.outlook = le_outlook.fit_transform(X.outlook)

le_temp = LabelEncoder()
X.temp = le_temp.fit_transform(X.temp)

le_humidity = LabelEncoder()
X.humidity = le_humidity.fit_transform(X.humidity)

le_windy = LabelEncoder()
X.windy = le_windy.fit_transform(X.windy)
print("\nNow the Train data is :\n",X.head())

le_PlayTennis = LabelEncoder()
y= le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20)
classifier=GaussianNB()
classifier.fit(X_train,y_train)
from sklearn.metrics import accuracy_score
print("Accuracy is:- ",accuracy_score(classifier.predict(X_test),y_test))
```

Output:-

```
PS C:\Users\Harsh\Desktop\ML> python -u "c:\Users\Harsh\Desktop\ML\NaiveBai.py"
The first 5 values of data is :
     outlook  temp humidity  windy playtenis
0     sunny   hot     high  False        no
1     sunny   hot     high   True        no
2  overcast   hot     high  False       yes
3     rainy  mild     high  False       yes
4     rainy  cool   normal  False       yes

The First 5 values of train data is
     outlook  temp humidity  windy
0     sunny   hot     high  False
1     sunny   hot     high   True
2  overcast   hot     high  False
3     rainy  mild     high  False
4     rainy  cool   normal  False

The first 5 values of Train output is
 0     no
1     no
2    yes
3    yes
4    yes
Name: playtenis, dtype: object

Now the Train data is :
    outlook  temp  humidity  windy
0        2     1         0      0
1        2     1         0      1
2        0     1         0      0
3        1     2         0      0
4        1     0         1      0

Now the Train output is
 [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
Accuracy is:-  0.3333333333333333
```