Problem Statement-1: Write a program to calculate the standard deviation and variance.

Objective: To understand how to find standard deviation and variation of grouped data without using library functions.

Input: A list of numbers and its size is taken from user.

Output: Standard deviation and variance of entered elements**.**

Source Code:

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

x = np.array([1, 2, 3, 4, 5, 8, 9, 12, 11, 10])
y = np.array([7, 14, 15, 18, 19, 20, 13, 10, 16, 17])

x_mean = np.mean(x)
y_mean = np.mean(y)
n = np.size(x)

x_mean, y_mean

var = np.sum(x*x)-n*x_mean*x_mean

print("variance is",var)

st_dev = np.sqrt(var)

print("standard deviation is",st_dev)
```

Output:

variance is 142.5
standard deviation is 11.937336386313323

Problem Statement-2: Write a program to implement the Find-S algorithm.

Objective: To understand how to find S algorithm.

Input: A csv file (enjoysport.csv).

Output: The total number of training instances, total number of num_attribute, initial hypothesis, hypothesis for the training instance and specific hypothesis for the training.

Source Code:

```
import pandas as pd
df = pd.read_csv(r"C:\Users\DELL\Downloads\enjoysport.csv")

import csv
a = []
with open(r"C:\Users\DELL\Downloads\enjoysport.csv","r") as csvfile:
    for row in csv.reader(csvfile):
        a.append(row)
        print("\n",a)
        print("---- ")

print("\n The total number of training instances are:",len(a)-1)
num_attribute = len(a[0])-1

print("\n The total number of num_attribute are:", len(a[0])-1)
print("\n The initial hypothesis is:")
hypothesis = ['0'] * num_attribute
print(hypothesis)

for i in range(0, len(a)):
    if a[i][num_attribute] == "yes":
        for j in range(0, num_attribute):
            if hypothesis[j] == "0" or hypothesis[j] == a[i][j]:
                hypothesis[j] = a[i][j]
            else:
                hypothesis[j]="?"
                print(hypothesis)

print(f"\n The hypothesis for the training instance {hypothesis} is:\n")
print("The specific hypothesis for the training instances is")
print(hypothesis)
```

Output:

[['sky', 'airtemp', 'humidity', 'wind', 'water', 'forcast', 'enjoysport']]

[['sky', 'airtemp', 'humidity', 'wind', 'water', 'forcast', 'enjoysport'], ['sunny', 'warm', 'normal', 'strong ', 'warm', 'same', 'yes']]

[['sky', 'airtemp', 'humidity', 'wind', 'water', 'forcast', 'enjoysport'], ['sunny', 'warm', 'normal', 'strong ', 'warm', 'same', 'yes'], ['sunny', 'warm', 'high', 'strong ', 'warm', 'same', 'yes']]

[['sky', 'airtemp', 'humidity', 'wind', 'water', 'forcast', 'enjoysport'], ['sunny', 'warm', 'normal', 'strong ', 'warm', 'same', 'yes'], ['sunny', 'warm', 'high', 'strong ', 'warm', 'same', 'yes'], ['rainy ', 'cold', 'high', ' strong ', 'warm', 'change', 'no']]

[['sky', 'airtemp', 'humidity', 'wind', 'water', 'forcast', 'enjoysport'], ['sunny', 'warm', 'normal', 'strong ', 'warm', 'same', 'yes'], ['sunny', 'warm', 'high', 'strong ', 'warm', 'same', 'yes'], ['rainy ', 'cold', 'high', ' strong ', 'warm', 'change', 'no'], ['sunny', 'warm', 'high', 'strong ', 'cool', 'change', 'yes']]

The total number of training instances are : 4

The total number of num_attribute are : 6

The initial hypothesis is :
['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 for the training instance 4 is :
['sunny', 'warm', '?', 'strong ', '?', '?']

The specific hypothesis for the training instance is
['sunny', 'warm', '?', 'strong ', '?', '?']

Problem Statement-3: Write a program to implement the candidate algorithm.

Objective: To understand the implementation of Candidate Elimination Algorithm.

Input: A csv file (enjoysport.csv).

sky	airtemp	humidity	wind	water	forcast	enjoysport
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	change	yes

Output: Steps of Candidate Elimination Algorithm, Final specfic hypothesis and

Final general hypothesis.

Source Code:

```
import csv
with open(r"C:\Users\DELL\Downloads\enjoysport.csv","r") as f:
    csv_file = csv.reader(f)
    data = list(csv_file)
    print(data)
    print("\n")
    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]="?"

        print("\n steps of candidate Elimination Algorithm", data.index(i)+1)
        print(s)
        print(g)

    gh = []
    for i in g:
        for j in i:
            if j!="?":
                gh.append(j)
```

```
print("\n Final specific hypothesis:\n",s)
print("\n Final general hypothesis:\n",gh)
```

Output:

```
[['sky', 'airtemp', 'humidity', 'wind', 'water', 'forcast', 'enjoysport'], ['sunny', 'warm', 'normal', 'strong ' , 'warm', 'same', 'yes'], ['sunny', 'warm', 'high', 'strong ', 'warm', 'same', 'yes'], ['rainy ', 'cold', 'high', 's trong ', 'warm', 'change', 'no'], ['sunny', 'warm', 'high', 'strong ', 'cool', 'change', 'yes']]


['sunny', 'warm', 'normal', 'strong ', 'warm', 'same']

Steps of Candidate Elimination Algorithm 1
['sunny', 'warm', 'normal', 'strong ', 'warm', 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Steps of Candidate Elimination Algorithm 2
['sunny', 'warm', 'normal', 'strong ', 'warm', 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Steps of Candidate Elimination Algorithm 3
['sunny', 'warm', '?', 'strong ', 'warm', 'same']


[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Steps of Candidate Elimination Algorithm 4
['sunny', 'warm', '?', 'strong ', 'warm', 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]

Steps of Candidate Elimination Algorithm 5
['sunny', 'warm', '?', 'strong ', '?', '?']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final specific hypothesis:
 ['sunny', 'warm', '?', 'strong ', '?', '?']

Final general hypothesis:
 [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

Problem Statement-4: Write a program to implement the linear regression using inbuilt function.

Objective: To understand the implementation of Linear Regression Algorithm.

Input: Elements of array**.**

Output: Slope, intercept, squared error, mean squared error, root mean square error, R square, graph plotted for linear regression.

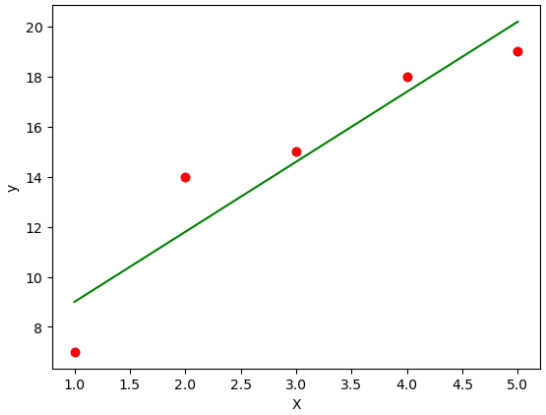
Source code -:

```
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)

x_mean, y_mean

Sxy = np.sum(x*y)-n*x_mean * y_mean # covariance(x,y)
Sxx = np.sum(x*x)-n*x_mean*x_mean  # variance(x)

b1 = Sxy / Sxx  # slope

b0 = y_mean - b1 * x_mean  # intercept

print("Slope b1 is", b1)
print("Intercept b0 is", b0)

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

y_pred = b1 * x + b0
y_pred

plt.scatter(x, y, color="yellow")
plt.plot(x, y_pred, color="red")
plt.xlabel("x")
plt.ylabel("y")

# Analyze the performance of the model by calculating mean squared error and Root mean Square error, R square

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

mse = se/n
```

```
print("mean squared error is", mse)

rmse = np.sqrt(mse)
print("root mean square error is", rmse)

SSt = np.sum((y - y_mean)**2)
R2 = 1-(se/SSt)
print("R square is",R2)
```

Output:

slope b1 is 2.8
intercept b0 is 6.200000000000001
squared error is 10.800000000000004
mean squared error is 2.160000000000001
root mean square error is 1.4696938456699071
R square is 0.8789237668161435



Problem Statement-5: Write a program to implement the linear regression without using library function.

Objective: To understand the implementation of Linear Regression Algorithm without numpy.

Input: A csv file (enjoysport.csv).

Output: Steps of Candidate Elimination Algorithm, Final specific hypothesis and Final general hypothesis.

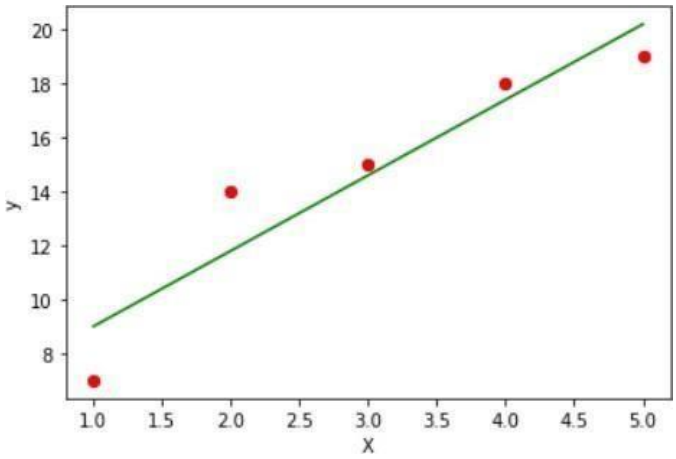
Source code -:

```
import matplotlib.pyplot as plt
import math
x = [1,2,3,4,5]
y = [7,14,15,18,19]
n = len(x)
sum_x=sum_y=x_mean=y_mean=se=SSt=0

 for k in range(0,n):
     y_mean = y_mean+y[k]
     x_mean =x_mean+x[k]
     sum_x=sum_x+x[k]*x[k]
     sum_y=sum_y+x[k]*y[k]

y_mean=y_mean/n
x_mean=x_mean/n
print(x_mean,y_mean)
Sxy = sum_y- n*x_mean*y_mean
Sxx = sum_x- n*x_mean*x_mean

b1 = Sxy/Sxx # m slope
b0 = y_mean-b1*x_mean # c intercept y=mx+c so c= y-mx
print('slope b1 is ', b1)
print('intercept b0 is ', b0)

plt.scatter(x,y)
plt.xlabel('Independent variable X')
plt.ylabel('Dependent variable y')
y_pred=[]
for i in  x:
      y_pred.append( b1 * i + b0)

plt.scatter(x, y, color = 'red')
plt.plot(x, y_pred, color = 'green')
plt.xlabel('X')
plt.ylabel('y')
error=[]
for i in range(0,len(y)):
error.append(y[i] - y_pred[i])
for i in error:
    se = se+(i**2)
```

```
print('squared error is', se)

mse = se/n


print('mean squared error is', mse)

rmse = math.sqrt(mse)
print('root mean square error is', rmse)
for i in range(0,len(y)):
    SSt = SSt+((y[i] - y_mean)**2)
R2 = 1- (se/SSt)
print('R square is', R2)
```

Output:

3.0 14.6
slope b1 is 2.8
intercept b0 is 6.200000000000001
squared error is 10.800000000000004
mean squared error is 2.160000000000001
root mean square error is 1.4696938456699071
R square is 0.8789237668161435



Problem Statement-6: Write a program to implement the KNN algorithm.

Objective: To understand the implementation of Knn Algorithm.

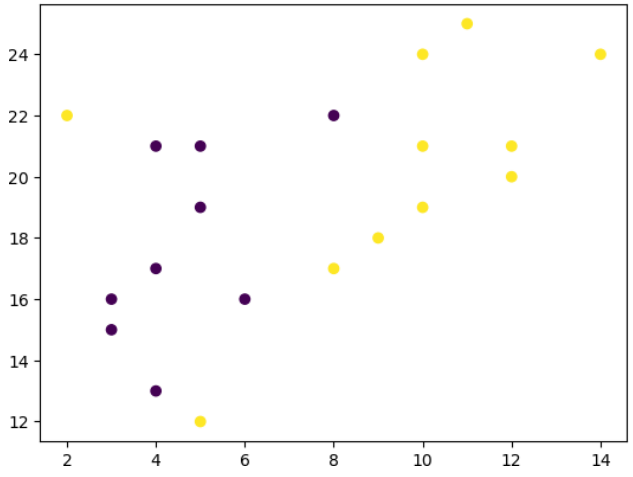
Input: List of attributes and labeled classes.
Output: Zipped data and different clusters along with their class shown in scatter graph.

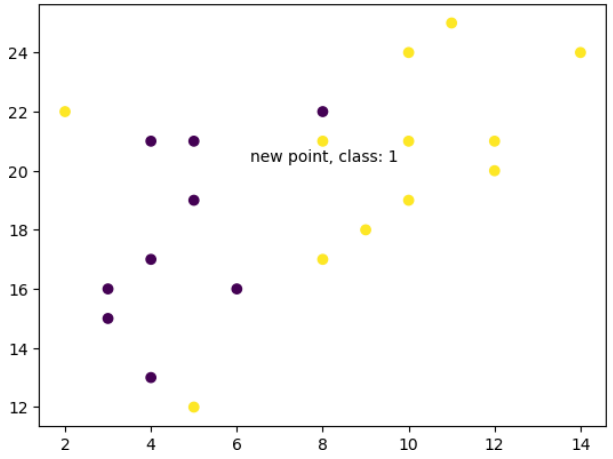
Source Code:

```
# knn Algo

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:

[(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)]





Problem Statement-7: Write a program to implement the K-mean Clustering algorithm.

Objective: To understand the implementation of K-Mean Clustering Algorithm**.**

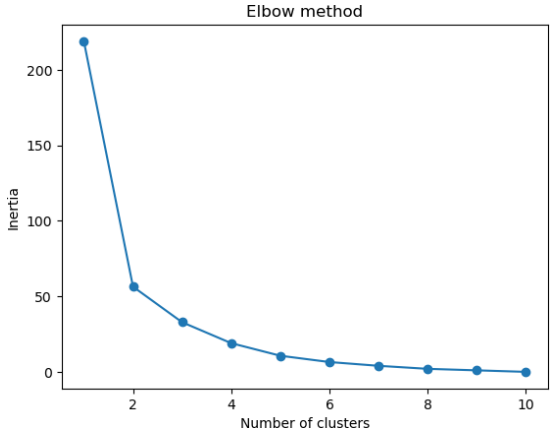
Input: List of attributes.

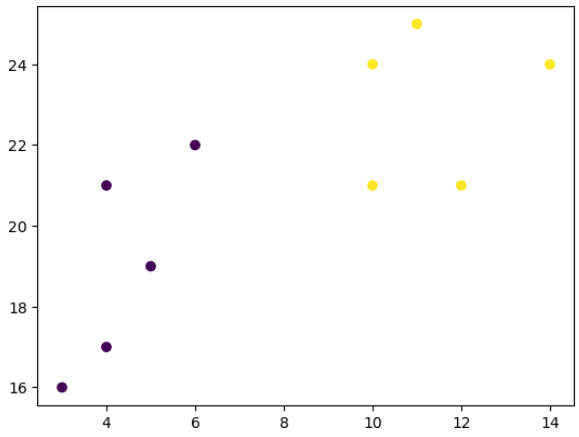
Output: Graph of elbow method showing relationship between inertia and number of clusters,scatter graph showing different clusters.

Source Code:

```
#K mean Clustering
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
data = list(zip(x, y))
print(data)
inertias = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)

plt.plot(range(1,11), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number 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:
[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (6, 22), (10, 21), (12, 21)]





Problem Statement-8: Write a program to implement the logistic regression.

Objective: To understand the implementation of Logistic Regression Algorithm**.**
Input: List of arrays.

Output: Prediction of categorical dependent variable and probabilistic values which lie between0 and 1.

Source Code:

```
import numpy as np
from sklearn import linear_model

# reshaped for logistic regression
X = np.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 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)

# predict for the size is 3.46mm:
predicted = logr.predict(np.array([3.46]).reshape(-1,1))
print("predicted value is",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:

```
[0]
[[0.60749955]
 [0.19268876]
 [0.12775886]
 [0.00955221]
 [0.08038616]
 [0.07345637]
 [0.88362743]
 [0.77901378]
 [0.88924409]
 [0.81293497]
 [0.57719129]
 [0.96664243]]
```

Problem Statement-9: Write a python program to implement Naïve Bayes Classifier.

Objective: To understand the implementation of Naïve Bayes Classifier.
Input: A csv file (play_tennis.csv).

	A	B	C	D	E	F
1	day	outlook	temp	humidity	wind	play
2	D1	Sunny	Hot	High	Weak	No
3	D2	Sunny	Hot	High	Strong	No
4	D3	Overcast	Hot	High	Weak	Yes
5	D4	Rain	Mild	High	Weak	Yes
6	D5	Rain	Cool	Normal	Weak	Yes
7	D6	Rain	Cool	Normal	Strong	No
8	D7	Overcast	Cool	Normal	Strong	Yes
9	D8	Sunny	Mild	High	Weak	No
10	D9	Sunny	Cool	Normal	Weak	Yes
11	D10	Rain	Mild	Normal	Weak	Yes
12	D11	Sunny	Mild	Normal	Strong	Yes
13	D12	Overcast	Mild	High	Strong	Yes
14	D13	Overcast	Hot	Normal	Weak	Yes
15	D14	Rain	Mild	High	Strong	No

Output: The first 5 values of data, first 5 values of train data, first 5 values of Train output,Train data (Target Variable, Features in Training set and accuracy score of algorithm.

Source Code:

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

data = pd.read_csv(r"C:\Users\DELL\Downloads\Data Set in csv format\play_tennis.csv")
data.drop("day", axis=1, inplace=True)

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

y = data.iloc[:,-1]
print("The First 5 Value of Train Output is")
print(y.head())

le_outlook = LabelEncoder()
X["outlook"] = le_outlook.fit_transform(X["outlook"])

le_temperature = LabelEncoder()
X["temp"] = le_temperature.fit_transform(X["temp"])

le_humidity = LabelEncoder()
X["humidity"] = le_humidity.fit_transform(X["humidity"])

le_wind = LabelEncoder()
X["wind"] = le_wind.fit_transform(X["wind"])

print("\nNow the Train data is")
print(X.head())

le_playtennis = LabelEncoder()
y = le_playtennis.fit_transform(y)

print("\n Now 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=0.2)

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:

```
The First 5 Values of data is:
    outlook  temp humidity    wind play
0     Sunny   Hot     High    Weak  No
1     Sunny   Hot     High  Strong  No
2  Overcast   Hot     High    Weak  Yes
3      Rain  Mild     High    Weak  Yes
4      Rain  Cool   Normal    Weak Yes
The First 5 Value of Train Output is
0     No
1     No
2    Yes
3    Yes
4    Yes
Name: play, dtype: object

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

 Now the train output is
 [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
Accuracy is  1.0
```

Problem Statement-10: Write a python program to implement ID3 Algorithm.

Objective: To understand the implementation of ID3 Algorithm.
Input: A csv file (play_tennis.csv).

	A	B	C	D	E	F
1	day	outlook	temp	humidity	wind	play
2	D1	Sunny	Hot	High	Weak	No
3	D2	Sunny	Hot	High	Strong	No
4	D3	Overcast	Hot	High	Weak	Yes
5	D4	Rain	Mild	High	Weak	Yes
6	D5	Rain	Cool	Normal	Weak	Yes
7	D6	Rain	Cool	Normal	Strong	No
8	D7	Overcast	Cool	Normal	Strong	Yes
9	D8	Sunny	Mild	High	Weak	No
10	D9	Sunny	Cool	Normal	Weak	Yes
11	D10	Rain	Mild	Normal	Weak	Yes
12	D11	Sunny	Mild	Normal	Strong	Yes
13	D12	Overcast	Mild	High	Strong	Yes
14	D13	Overcast	Hot	Normal	Weak	Yes
15	D14	Rain	Mild	High	Strong	No

Output: The first 5 values of data, first 5 values of train data, first 5 values of Train output, Train data (Target Variable, Features in Training set and accuracy score of algorithm.

Source Code:

```
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(r"C:\Users\DELL\Downloads\Data Set in csv format\play_tennis.csv")
data.drop("day", axis=1, inplace=True)

print("The First 5 Value of data is")
print(data.head())

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

le_outlook = LabelEncoder()
X["outlook"] = le_outlook.fit_transform(X["outlook"])

le_temperature = LabelEncoder()
X["temp"] = le_temperature.fit_transform(X["temp"])

le_humidity = LabelEncoder()
X["humidity"] = le_humidity.fit_transform(X["humidity"])

le_wind = LabelEncoder()
X["wind"] = le_wind.fit_transform(X["wind"])

print("\nNow the Train data is")
print(X.head())

le_playtennis = LabelEncoder()
```

```
y = le_playtennis.fit_transform(y)

print("\n Now the train data (Traget Variable) is\n",y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
print("\nFeatures in Training set is:\n",X_train)

print("\n Test Set is:\n",X_test)

classifier = DecisionTreeClassifier(criterion="entropy")
classifier.fit(X_train,y_train)

pred = classifier.predict(X_test)
print(f"\n for input \n {X_test}, \n we obtain {le_playtennis.inverse_transform(pred)}")
print("\n Accuracy Score is:", metrics.accuracy_score(y_test, pred))
```

Output:

```
The First 5 Value of data is
    outlook  temp humidity    wind play
0     Sunny   Hot     High    Weak  No
1     Sunny   Hot     High  Strong  No
2  Overcast   Hot     High    Weak  Yes
3      Rain  Mild     High    Weak  Yes
4      Rain  Cool   Normal    Weak  Yes

 The First 5 values of Train output is
    outlook  temp humidity    wind
0     Sunny   Hot     High    Weak
1     Sunny   Hot     High  Strong
2  Overcast   Hot     High    Weak
3      Rain  Mild     High    Weak
4      Rain  Cool   Normal    Weak

 The First 5 values of Train output is
 0     No
1     No
2    Yes
3    Yes
4    Yes
Name: play, dtype: object

Now the Train data is
   outlook temp humidity  wind
0       2    1        0     1

1       2    1        0     0
2       0    1        0     1
3       1    2        0     1
4       1    0        1     1

 Now the train data (Traget Variable) is
 [0 0 1 1 1 0 1 0 1 1 1 1 1 0]

Features in Training set is:
    outlook  temp  humidity  wind
0        2     1        0     1
2        0     1        0     1
13       1     2        0     0
3        1     2        0     1
4        1     0        1     1
6        0     0        1     0
10       2     2        1     0
8        2     0        1     1
11       0     2        0     0
12       0     1        1     1
1        2     1        0     0

 Test Set is:
   outlook  temp  humidity  wind
```

9 1 2 1 1
5 1 0 1 0
7 2 2 0 1

for input
outlook temp humidity wind
9 1 2 1 1
5 1 0 1 0
7 2 2 0 1,
we obtain ['Yes' 'Yes' 'Yes']

Accuracy Score is: 0.3333333333333333