* MACHINE LEARNING ALGORITHMS *

1. Implement and demonstrate FIND-S algorithm for finding the most specific hypothesis based on given set of training data samples. Read the training data from a .csv file.

Description :-

The Find-S algorithm is a basic concept-learning algorithm used in machine learning. It starts with the most specific hypothesis, which is an empty hypothesis that contains no attribute or value. It then considers each training example and generalizes the hypothesis by adding the attribute values that are consistent with the example. The algorithm stops when all the training examples are consistent with the hypothesis, or when there are no more examples to consider. The resulting hypothesis is the most specific hypothesis that is consistent with the training data.

Dataset:-

	sky	airtemp	humidity	wind	water	forecar	enjoysport
0	sunny	warm	normal	strong	warm	same	yes
1	sunny	warm	high	strong	warm	same	yes
2	Rain	cold	high	strong	warm	change	no
3	Sunny	warm	high	strong	cool	change	yes

Python Code:-

```
import pandas as pd

data=pd.read_csv("C:/Users/ramya/OneDrive/Documents/ml_csv/finds.csv")

print(data)

h0=[]

d=0
```

```
for i in enumerate(data):
    d=d+1
for i in range(d-1):
    h0.append(0)
print("hypothesis initially:",h0)
count=0
```

output:-

```
sky       airtemp   humidity    wind       water    forecar   enjoysport
 sunny    warm     normal      strong     warm     same         yes
 sunny    warm      high       strong      warm     same         yes
  rain     cold      high          strong     warm    change       no
sunny    warm       high        strong      cool     change       yes

hypothesis initially: [0, 0, 0, 0, 0, 0]
```

```
for i in range(len(data)):
    if(data.iloc[i,-1]=='yes'):
        count=count+1
        #print(count)
        for j in range(len(h0)):
            a=data.iloc[i,j]
            if(h0[j]==a):
                continue
            elif(count==1):
                h0[j]=a
                #print(h0[j])
            else:
                h0[j]='?'
        print(h0)
```

output:-

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

2. For a given set of training data examples stored in a .csv file, implement and demonstrate thecandidate elimination algorithm to output a description of the set of all hypotheses consistent withthe training examples.

Description :-

The Candidate Elimination algorithm is a concept-learning algorithm used in machine learning. It starts with the most specific and most general hypotheses, which are the hypotheses that contain the smallest and largest possible sets of attribute values, respectively. It then considers each training example and updates the hypotheses by eliminating inconsistent hypotheses and generalizing or specializing the remaining hypotheses as necessary. The algorithm stops when it finds a single hypothesis that fits all the training examples or when it cannot further generalize or specialize the hypotheses. The resulting set of hypotheses is a minimal set of hypotheses that are consistent with the training data.

Dataset :-

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

Python Code :-

```
import pandas as pd

data=pd.read_csv("C:/Users/ramya/OneDrive/Documents/ml_csv/finds.csv")

d=0

count=0

g=[]
```

```
s=[]
for i in enumerate(data):
    d=d+1
g= [["?" for i in range(d-1)] for i in range(d-1)]
for i in range(d-1):
    s.append(data.iloc[1,i])
for i in range(len(data)):
    if(data.iloc[i,-1]=='yes'):
        for j in range(d-1):
            if(s[j]==data.iloc[i,j]):
                continue
            else:
                s[j]="?"
        print(“Specific : ”,s)
    else:
        for j in range(d-1):
            if(s[j]!=data.iloc[i,j]):
                g[j][j]=s[j]
            else:
                continue
        print(“Generalized : ”,g)
for i in range(d-1):
    if(g[i][i]!=s[i]):
        g[i][i]='?'

indices = [i for i, val in enumerate(g) if val == ['?', '?', '?', '?', '?', '?']]
for i in indices:
    g.remove(['?', '?', '?', '?', '?', '?'])
print("Synchronization : ",g)
```

Output :-

```
Specific :  ['sunny', 'warm', '?', 'strong', 'warm', 'same']

Specific :  ['sunny', 'warm', '?', 'strong', 'warm', 'same']

Generalized :  [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?
', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'sam
e']]

Specific :  ['sunny', 'warm', '?', 'strong', '?', '?']

Synchronization : [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?
', '?', '?', '?']]
```

3.Linear and Multilinear Regression on two different .csv Files.

Description :-

Linear Regression :-

Linear regression is a statistical method that models the relationship between a dependent variable and one independent variable by fitting a straight line to the data.

$y= a_0+a_1x+ \varepsilon$

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

a0= intercept of the line (Gives an additional degree of freedom)

a1 = Linear regression coefficient (scale factor to each input value).

ε = random error

Dataset :-

	attendence	marks
0	67	69
1	68	70

attendence	marks	
2	69	71
3	70	72
4	71	73
5	72	74
6	73	75
7	74	76
8	75	77
9	76	78
10	77	79
11	78	80
12	79	81
13	80	82
14	81	83
15	82	84
16	83	85
17	84	86
18	85	87
19	86	88

Python Code :-

```
import pandas as pd
```

```
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as mp
data=pd.read_csv("C:/Users/ramya/OneDrive/Documents/ml_csv/linear.csv")
print(data)
x=data.iloc[:,0:-1]
y=data.iloc[:,-1]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
model=LinearRegression()
model.fit(x_train,y_train)
result=model.predict(x_test)
result
print(r2_score(y_test,result))
```

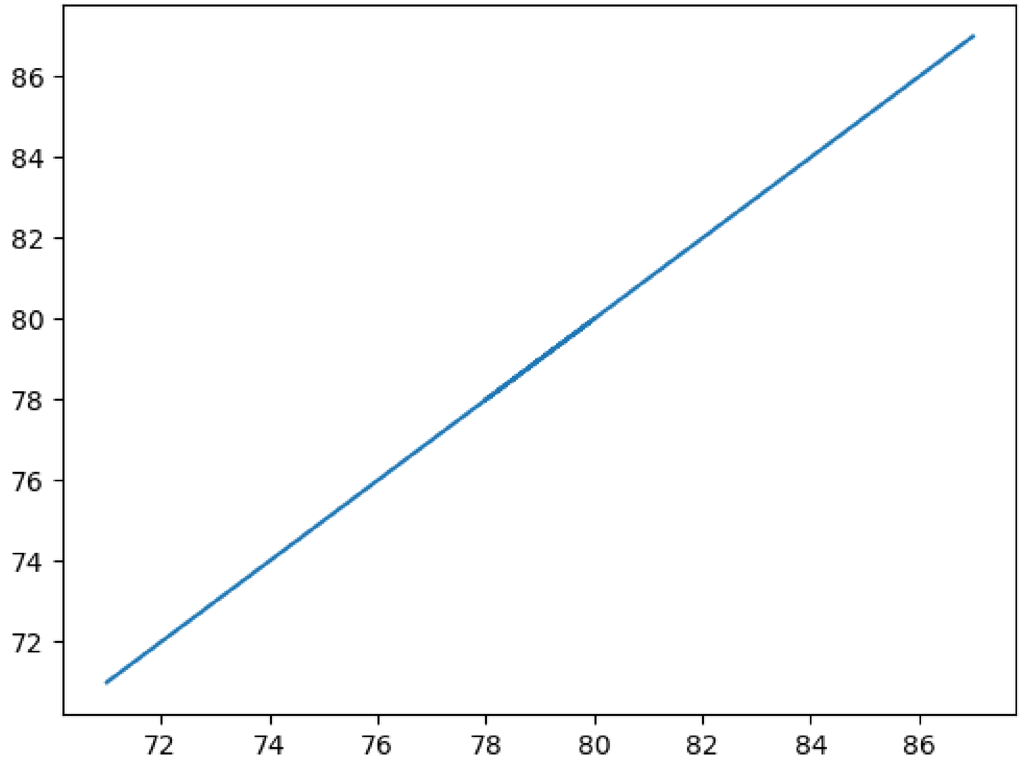
Output :-

Data :-

```
       Attendance  marks
0              67     69
1              68     70
2              69     71
3              70     72
4              71     73
5              72     74
6              73     75
7              74     76
8              75     77
9              76     78
10             77     79
11             78     80
12             79     81
13             80     82
14             81     83
15             82     84
16             83     85
17             84     86
18             85     87
```

```
19          86     88

Accuracy :1.0
```

mp. plot(y_test,result)



Multilinear Regression:-

Description :-

Multilinear regression is an extension of linear regression that models the relationship between a dependent variable and multiple independent variables by fitting a hyperplane to the data.

$yi=\beta 0+\beta 1xi1+\beta 2xi2+...+\beta pxip+\epsilon$

where, for $i=n$ **observations:**

yi=dependent variable

xi=explanatory variables

$\beta 0$=y-intercept (constant term)

βp=slope coefficients for each explanatory variable

ϵ=the model’s error term (also known as the residuals)

Dataset : -

attendence	certification	marks	
0	65	2	68
1	66	3	69
2	67	3	70
3	68	2	71
4	69	3	72
5	70	2	73
6	71	4	74
7	72	2	75
8	73	3	76
9	74	3	77
10	75	2	78
11	76	1	79
12	77	2	80
13	78	2	81

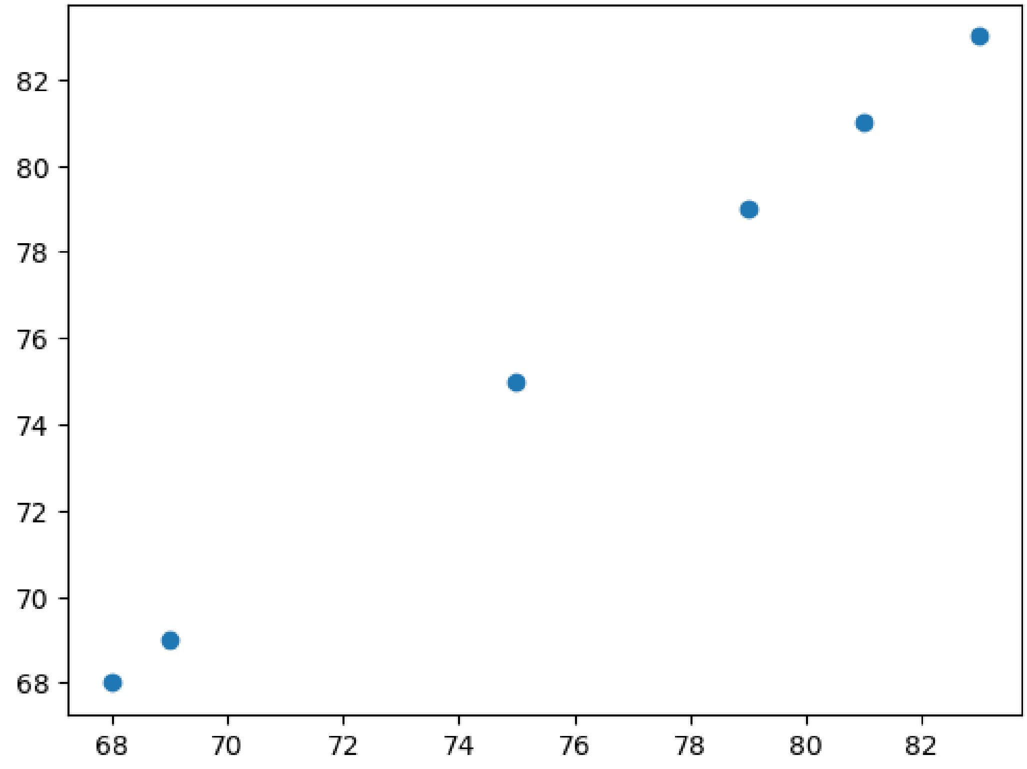
attendence	certification	marks	
14	79	2	82
15	80	2	83
16	81	2	84
17	82	2	85
18	83	1	86
19	84	2	87

Python Code:-

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
data=pd.read_csv("C:/Users/ramya/OneDrive/Documents/ml_csv/multi.csv")
print(data)
x=data.iloc[:,0:-1]
y=data.iloc[:,-1]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
model=LinearRegression()
model.fit(x_train,y_train)
result=model.predict(x_test)
result
print(r2_score(y_test,result))
```

```
    attendence  certification   marks
```

```
0           65              2     68
1           66              3     69
2           67              3     70
3           68              2     71
4           69              3     72
5           70              2     73
6           71              4     74
7           72              2     75
8           73              3     76
9           74              3     77
10          75              2     78
11          76              1     79
12          77              2     80
13          78              2     81
14          79              2     82
15          80              2     83
16          81              2     84
17           82              2     85
18           83              1     86
19           84              2     87

Accuracy :1.0
```



4.Polynomial Regression on student marks data

Description :-

Polynomial regression is a type of linear regression in which the relationship between the dependent variable and one or more independent variables is modeled as an nth-degree polynomial. The technique allows for a more flexible model that can capture nonlinear relationships between the variables. The degree of the polynomial can be chosen based on the complexity of the relationship between the variables, but higher degrees can lead to overfitting. The parameters of the polynomial regression model can be estimated using methods such as the least squares method or maximum likelihood estimation. Once the parameters are estimated, they can be used to make predictions about the dependent variable based on the independent variables.

$$f(x) = c_0 + c_1\, x + c_2\, x^2 \cdots c_n\, x^n$$

Dataset :-

attendence	certification	marks	
0	65	2	68
1	66	3	69
2	67	3	70
3	68	2	71
4	69	3	72
5	70	2	73
6	71	4	74
7	72	2	75
8	73	3	76

attendence	certification	marks	
9	74	3	77
10	75	2	78
11	76	1	79
12	77	2	80
13	78	2	81
14	79	2	82
15	80	2	83
16	81	2	84
17	82	2	85
18	83	1	86
19	84	2	87

Python Code :-

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import  LinearRegression
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
data=pd.read_csv("C:/Users/ramya/OneDrive/Documents/ml_csv/multi.csv")
x=data.drop("marks",axis=1)
y=data[["marks"]]
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

p=PolynomialFeatures(degree=3)

x_train_poly=p.fit_transform(x_train)

x_test_poly=p.fit_transform(x_test)

print(x_train)

print(x_train_poly)

print(x_test)

print(x_test_poly)
```

Output : #Train

```
        attendence     certification
6       71          4
15      80          2
5       70          2
7       72          2
10      75          2
14      79          2
12      77          2
1       66          3
18      83          1
4       69          3
19      84          2
13      78          2
16      81          2
11      76          1
```

#after polynomial degree applied Train

```
[[1.00000e+00 7.10000e+01 4.00000e+00 5.04100e+03 2.84000e+02 1.60000e+01
  3.57911e+05 2.01640e+04 1.13600e+03 6.40000e+01
```

#Test

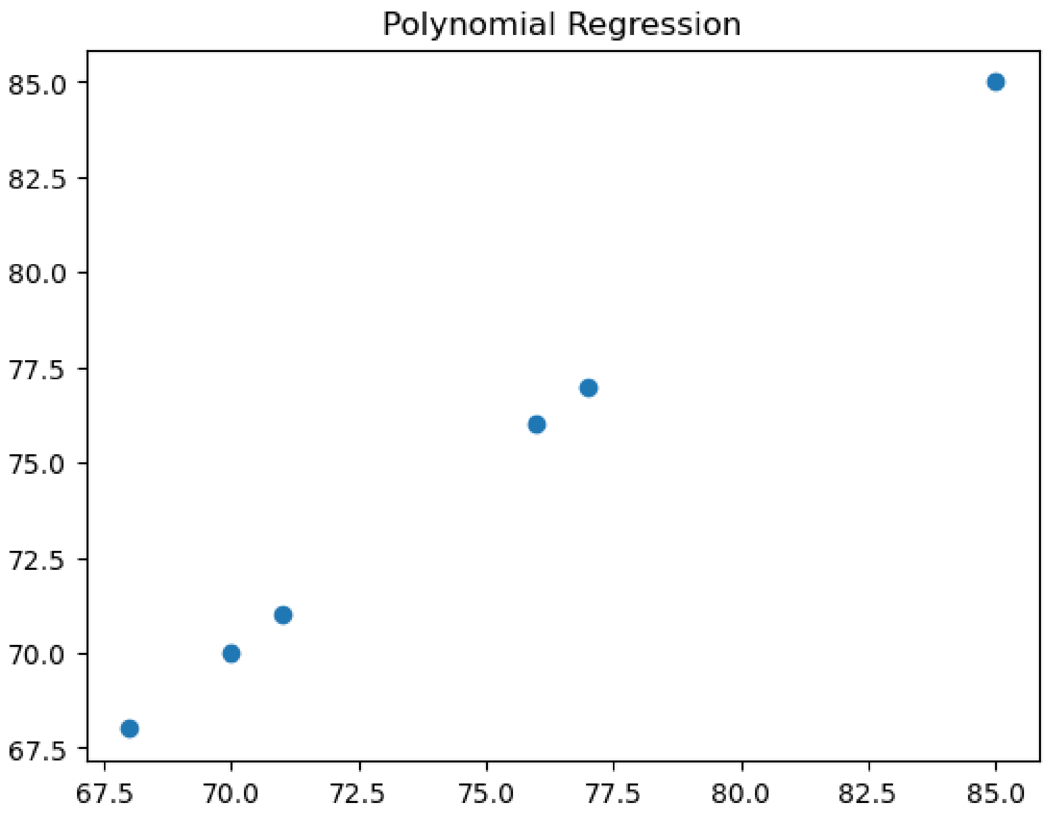
```
     attendence   certification
17      82          2
0       65          2
8       73          3
2       67          3
9       74          3
3       68          2
```

#After polynomial degree Aplied test

```
[[1.00000e+00 8.20000e+01 2.00000e+00 6.72400e+03 1.64000e+02 4.00000e+00
  5.51368e+05 1.34480e+04 3.28000e+02 8.00000e+00]
 . . .
```

```
model=LinearRegression()

model.fit(x_train_poly,y_train)

y_predict=model.predict(x_test_poly)

acc=r2_score(y_test,y_predict)

print(acc)

plt.title("Polynomial Regression")

plt.scatter(y_test,y_predict)
```

```
accuracy: 0.9999999997846436
```



5. Write a program to demonstrate the working of Logistic Regression classifier. Use appropriate dataset for Logistic Regression.

Evalution Parameters : f1_score,accuracy,recall

Description :-

Logistic regression is a statistical technique used to model the relationship between a binary dependent variable and one or more independent variables. The technique estimates the probability of the dependent variable taking a particular value (e.g., 0 or 1) based on the values of the independent variables. The model uses a logistic function to transform the linear regression equation into a probability score between 0 and 1

Dataset :-

https://github.com/plotly/datasets/blob/master/diabetes.csv

Python code:-

```
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

data=pd.read_csv("C:/Users/ramya/OneDrive/Documents/ml_csv/diabetes/diabetes.csv")

df=data.isnull().sum()

#print(df)

x=data.drop("Outcome",axis=1)

y=data[["Outcome"]]

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

print(x)

print(y)
```

```
Pregnancies                 0
```

```
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
```

```
st=StandardScaler()

x_train=st.fit_transform(x_train)

x_test=st.fit_transform(x_test)

model=LogisticRegression()

model.fit(x_train,y_train)

y_predict=model.predict(x_test)

print("predicted:",y_predict)

acc=accuracy_score(y_test,y_predict)

print("accuracy : ",acc)

print(y_test)

from sklearn import metrics

confusion_matrix = metrics.confusion_matrix(y_test,y_predict)

print(confusion_matrix)
```

```
#Tp,Fp,Tn,Fn output:-

[[138  26]
 [ 25  42]]
```

```
print("f1score: ",f1_score(y_test,y_predict))

print("recall_score",recall_score(y_test,y_predict))

print("precission",precision_score(y_test,y_predict))
```

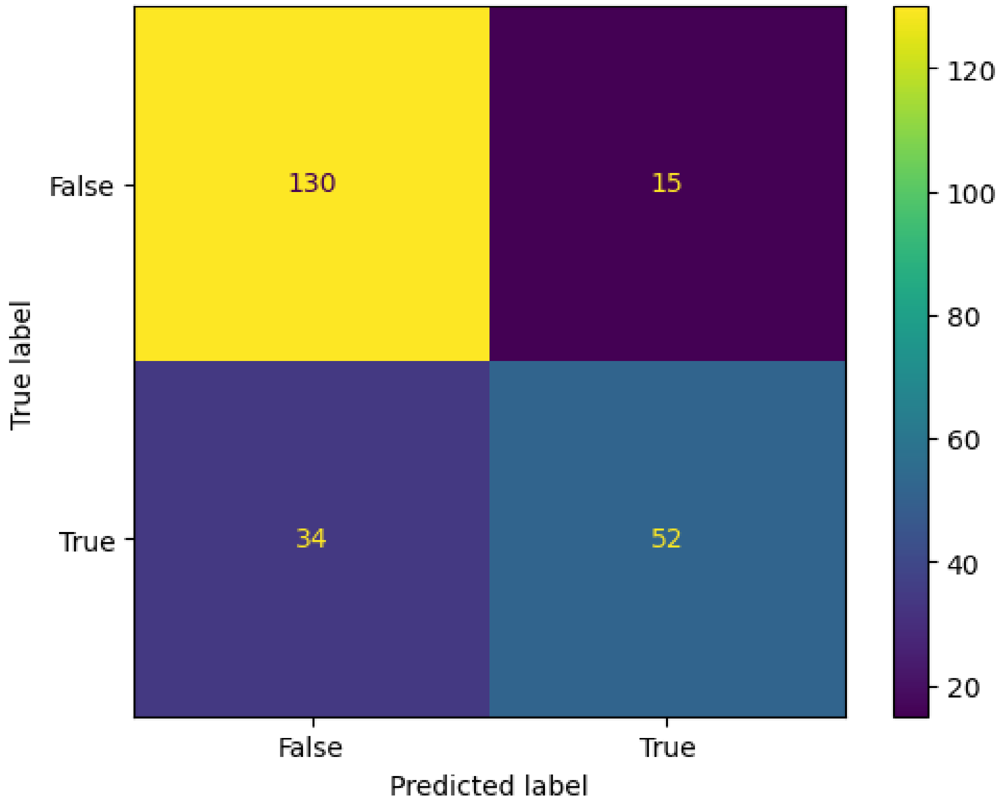
output :-

```
f1score:  0.6222222222222222
recall_score 0.6268656716417911
precission 0.6176470588235294
```

```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_
matrix, display_labels = [False, True])
cm_display.plot()
plt.show()
```

Output:-

Visualization with Confusion matrix



6. Write a program to demonstrate the working of the decision tree classifier. Use appropriate dataset for building the decision tree and apply this knowledge to classify a new sample.

Description :-

The Decision Tree Classifier is a machine learning algorithm that uses a tree-like model of decisions and their possible consequences to classify input data. It partitions the input data based on the values of the input features and recursively splits the data into subsets that are as homogeneous as possible in terms of the target variable. The resulting tree is then used to predict the target variable for new data points.

Dataset : -

https://github.com/plotly/datasets/blob/master/diabetes.csv

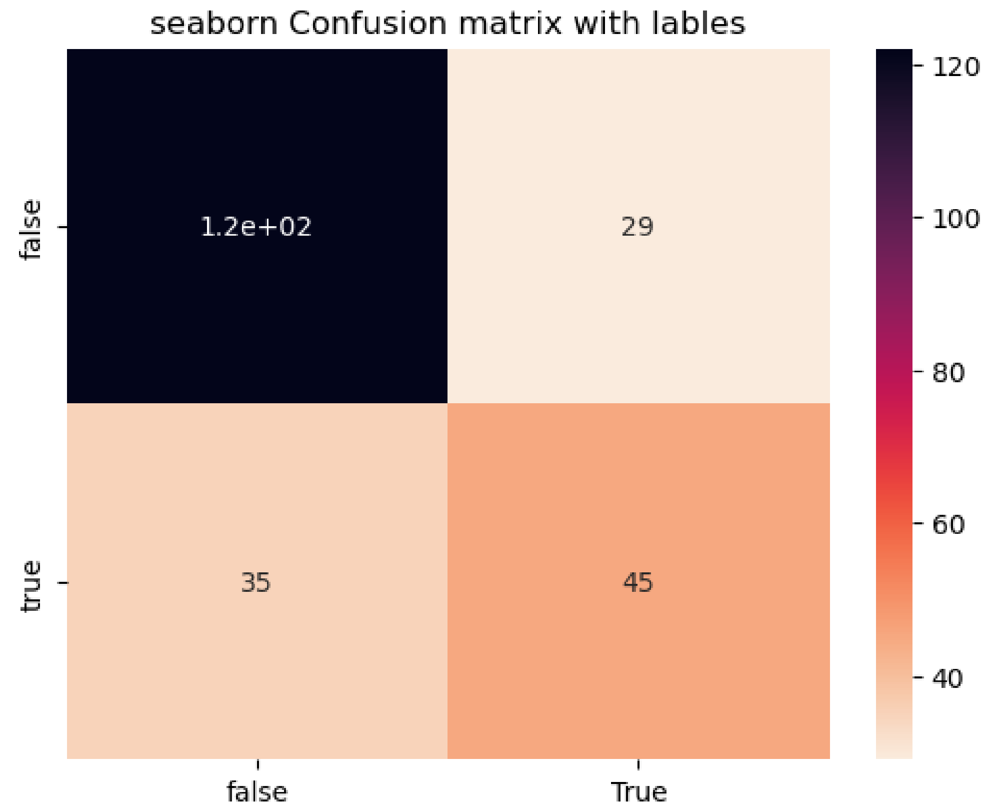
Python Code : -

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score,confusion_matrix
import seaborn as sns
from sklearn.preprocessing import StandardScaler
df=pd.read_csv("C:/Users/ramya/OneDrive/Documents/ml_csv/diabetes/diabetes.csv")
print(df)
x=df.drop("Outcome",axis=1)
y=df["Outcome"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
st=StandardScaler()
#print(x_train)
#print(x_test)
```

```
x_train=st.fit_transform(x_train)
x_test=st.fit_transform(x_test)
#print(x_train)
#print(x_test)
model=DecisionTreeClassifier(criterion='gini')
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
accu=accuracy_score(y_test,y_pred)
print("accuracy:",accu)
cnm=confusion_matrix(y_test, y_pred)
a=sns.heatmap(cnm,annot=True,cmap='rocket_r')
a.set_title("seaborn Confusion matrix with lables")
a.xaxis.set_ticklabels(["false","True"])
a.yaxis.set_ticklabels(["false","true"])
features=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI',
'DiabetesPedigreeFunction','Age']
```

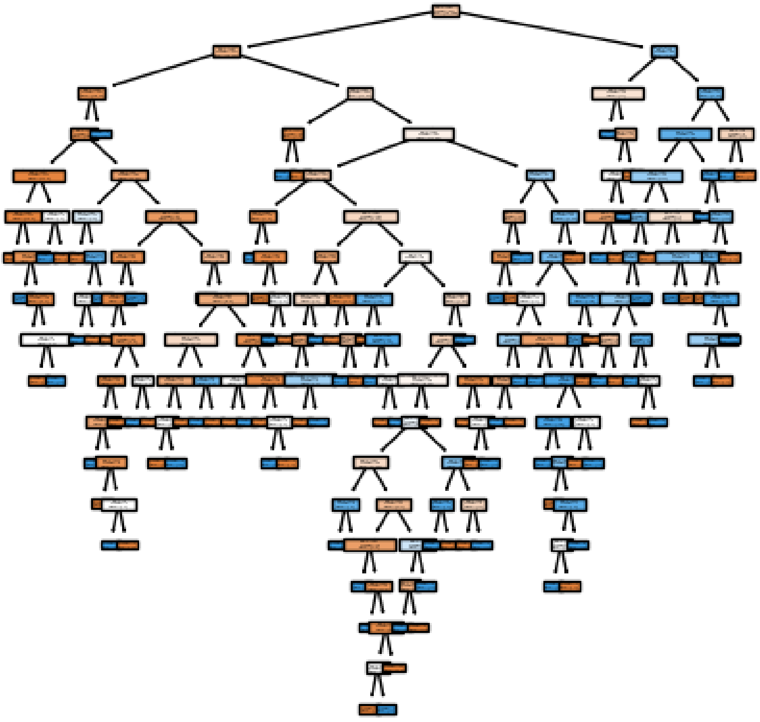
Output:-

```
accuracy: 0.7229437229437229
```



```
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5))
plot_tree(model,filled=True,feature_names=features)
plt.show()
```

Output



7.Write a program to demonstrate the working of Decision tree regressor. Use appropriate dataset for decision tree regressor.

Description :-

The Decision Tree Regressor is a machine learning algorithm used for regression tasks. It works by recursively partitioning the input space into smaller and smaller regions, and then predicting the output value for each region based on the mean or median of the training data in that region.

Dataset :-

Student Marks dataset

Python Code :-

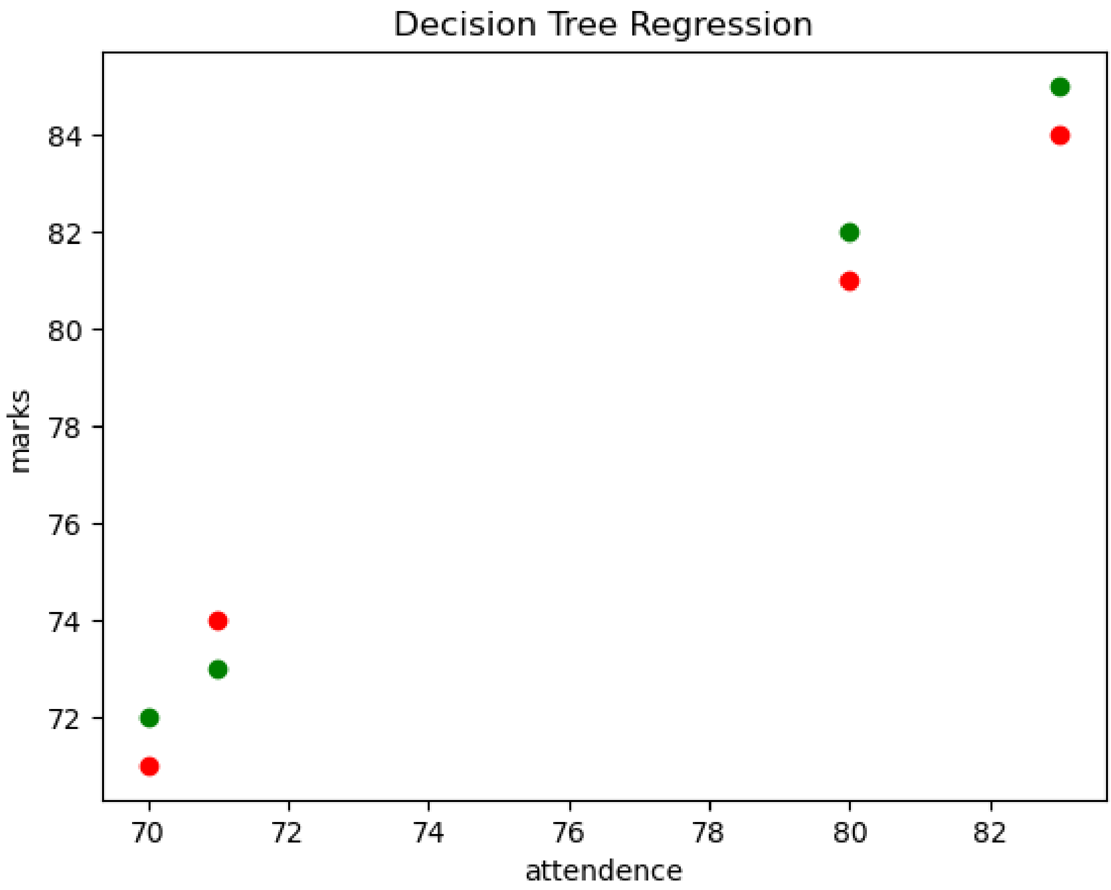
```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor

import matplotlib.pyplot as plt

#DecisionTreeRegressor
```

```
from sklearn.metrics import r2_score
d=pd.read_csv("C:/Users/ramya/OneDrive/Documents/ml_csv/linear.csv")
print(d)
x=d[["attendence"]]
y=d["marks"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
model=DecisionTreeRegressor()
model.fit(x_train,y_train)
r=model.predict(x_test)
y=r2_score(y_test,r)
print(y)
plt.scatter(x_test,r,color="red")
plt.scatter(x_test,y_test,color="green")
plt.title('Decision Tree Regression')
plt.xlabel('attendence')
plt.ylabel('marks')
plt.show()
```

Output :-

```
       attendence  marks
0           67     69
1           68     70
2           69     71
3           70     72
4           71     73
5           72     74
6           73     75
7           74     76
8           75     77
9           76     78
10          77     79
11          78     80
12          79     81
```

```
13          80      82
14          81      83
15          82      84
16          83      85
17          84      86
18          85      87
19          86      88


Accuracy : 0.9682539682539683
```



8. Data PreProcessing and correlation between salnity and temperature

Description :-

Data preprocessing is the process of transforming raw data into a format that can be easily understood and analyzed by machine learning models. This involves cleaning, normalization, scaling, encoding, and feature selection.

Dataset :-

C:\Users\ML Lab\Downloads\bottle.csv

Program:

```
import pandas as pd

b=pd.read_csv("C:\\Users\\ML Lab\\Downloads\\bottle.csv\\bottle.csv")

print(b)
```

Output:

```
Cst_Cnt  Btl_Cnt      Sta_ID                              Depth_ID  \
0       1       1  054.0 056.0  19-4903CR-HY-060-0930-05400560-0000A-3
1       1       2  054.0 056.0  19-4903CR-HY-060-0930-05400560-0008A-3
2       1       3  054.0 056.0  19-4903CR-HY-060-0930-05400560-0010A-7
3       1       4  054.0 056.0  19-4903CR-HY-060-0930-05400560-0019A-3
4       1       5  054.0 056.0  19-4903CR-HY-060-0930-05400560-0020A-7

   Depthm  T_degC  Salnty  O2ml_L  STheta  O2Sat  ...  R_PHAEO  R_PRES  \
0       0   10.50  33.440     NaN  25.649    NaN  ...      NaN       0
1       8   10.46  33.440     NaN  25.656    NaN  ...      NaN       8
2      10   10.46  33.437     NaN  25.654    NaN  ...      NaN      10
3      19   10.45  33.420     NaN  25.643    NaN  ...      NaN      19
4      20   10.45  33.421     NaN  25.643    NaN  ...      NaN      20

   R_SAMP  DIC1  DIC2  TA1  TA2  pH2  pH1  DIC Quality Comment
0     NaN   NaN   NaN  NaN  NaN  NaN  NaN                  NaN
1     NaN   NaN   NaN  NaN  NaN  NaN  NaN                  NaN
2     NaN   NaN   NaN  NaN  NaN  NaN  NaN                  NaN
3     NaN   NaN   NaN  NaN  NaN  NaN  NaN                  NaN
4     NaN   NaN   NaN  NaN  NaN  NaN  NaN                  NaN

[5 rows x 74 columns]b=b.dropna(axis=1,thresh=800000)
```

```
print(b)
```

<u>Output:</u>

```
Cst_Cnt  Btl_Cnt       Sta_ID                            Depth_ID  \
0           1        1  054.0 056.0  19-4903CR-HY-060-0930-05400560-0000A-3
1           1        2  054.0 056.0  19-4903CR-HY-060-0930-05400560-0008A-3
2           1        3  054.0 056.0  19-4903CR-HY-060-0930-05400560-0010A-7
3           1        4  054.0 056.0  19-4903CR-HY-060-0930-05400560-0019A-3
4           1        5  054.0 056.0  19-4903CR-HY-060-0930-05400560-0020A-7
...        ...      ...          ...                                 ...
864858    34404   864859  093.4 026.4  20-1611SR-MX-310-2239-09340264-0
000A-7
864859    34404   864860  093.4 026.4  20-1611SR-MX-310-2239-09340264-0
002A-3
864860    34404   864861  093.4 026.4  20-1611SR-MX-310-2239-09340264-0
005A-3
864861    34404   864862  093.4 026.4  20-1611SR-MX-310-2239-09340264-0
010A-3
864862    34404   864863  093.4 026.4  20-1611SR-MX-310-2239-09340264-0
015A-3

        Depthm  T_degC   Salnty    STheta  RecInd  T_prec  ...  DarkAq  \
0            0  10.500  33.4400  25.64900       3     1.0  ...     9.0
1            8  10.460  33.4400  25.65600       3     2.0  ...     9.0
2           10  10.460  33.4370  25.65400       7     2.0  ...     9.0
3           19  10.450  33.4200  25.64300       3     2.0  ...     9.0
4           20  10.450  33.4210  25.64300       7     2.0  ...     9.0
...        ...     ...      ...       ...     ...     ...  ...     ...
864858       0  18.744  33.4083  23.87055       7     2.0  ...     9.0
864859       2  18.744  33.4083  23.87072       3     2.0  ...     9.0
864860       5  18.692  33.4150  23.88911       3     2.0  ...     9.0
864861      10  18.161  33.4062  24.01426       3     2.0  ...     9.0
864862      15  17.533  33.3880  24.15297       3     2.0  ...     9.0

        MeanAq  R_Depth  R_TEMP  R_POTEMP  R_SALINITY  R_SIGMA  R
_SVA  \
0          9.0        0   10.50     10.50      33.440   25.640  233.0
1          9.0        8   10.46     10.46      33.440   25.650  232.5
2          9.0       10   10.46     10.46      33.437   25.650  232.8
3          9.0       19   10.45     10.45      33.420   25.640  234.1
4          9.0       20   10.45     10.45      33.421   25.640  234.0
...        ...      ...     ...       ...         ...      ...    ...
864858     9.0        0   18.74     18.74      33.408   23.871  402.4
```

```
864859    9.0       2   18.74     18.74      33.408   23.871  402.5
864860    9.0       5   18.69     18.69      33.415   23.889  400.8
864861    9.0      10   18.16     18.16      33.406   24.014  389.1
864862    9.0      15   17.53     17.53      33.388   24.153  376.0

       R_DYNHT  R_PRES
0        0.000       0
1        0.010       8
2        0.020      10
3        0.040      19
4        0.040      20
...        ...     ...
864858   0.000       0
864859   0.008       2
864860   0.020       5
864861   0.040      10
864862   0.059      15

[864863 rows x 24 columns]
```

———

b.info()

Output:

```
-
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 24 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   Cst_Cnt     864863 non-null  int64
 1   Btl_Cnt     864863 non-null  int64
 2   Sta_ID      864863 non-null  object
 3   Depth_ID    864863 non-null  object
 4   Depthm      864863 non-null  int64
 5   T_degC      853900 non-null  float64
 6   Salnty      817509 non-null  float64
 7   STheta      812174 non-null  float64
 8   RecInd      864863 non-null  int64
 9   T_prec      853900 non-null  float64
 10  S_prec      817509 non-null  float64
 11  NH3q        808299 non-null  float64
 12  C14A1q      848605 non-null  float64
 13  C14A2q      848623 non-null  float64
 14  DarkAq      840440 non-null  float64
 15  MeanAq      840439 non-null  float64
 16  R_Depth     864863 non-null  int64
 17  R_TEMP      853900 non-null  float64
 18  R_POTEMP    818816 non-null  float64
 19  R_SALINITY  817509 non-null  float64
 20  R_SIGMA     812007 non-null  float64
 21  R_SVA       812092 non-null  float64
 22  R_DYNHT     818206 non-null  float64
 23  R_PRES      864863 non-null  int64
dtypes: float64(16), int64(6), object(2)
memory usage: 158.4+ MB
```

```
import numpy as np
mean_value=b["T_degC"].mean()
b["T_degC"].fillna(value=mean_value, inplace=True)

e=b["T_degC"].isnull().sum()
```

```
print(e)
```

Output:

```
0
```

```
import numpy as np
mean_value=b["Salnty"].mean()
b["Salnty"].fillna(value=mean_value, inplace=True)

a=b["Salnty"].isnull().sum()
print(a)
```

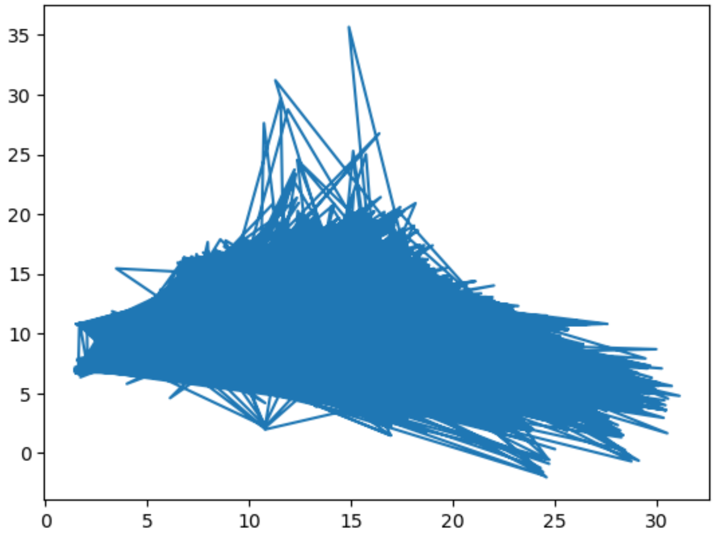
Output:

```
0
```

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.model_selection import train_test_split
x=b[["Salnty"]]
y=b['T_degC']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_test)
model=LinearRegression()
model.fit(x_train,y_train)
y_prediction=model.predict(x_test)
print(y_prediction)
import matplotlib.pyplot as plt
p=plt.plot(y_test,y_prediction)
print(p)
```

Output:

```
         Salnty
683571  34.13300
415965  33.84035
615594  33.98800
188585  33.89000
685504  33.42600
...          ...
619379  33.99100
771395  34.18900
737511  33.16000
521783  34.35700
473635  33.84035

[259459 rows x 1 columns]
[ 9.45279419 10.79847488 10.11954251 ... 13.92690535  8.42278299
 10.79847488]
[<matplotlib.lines.Line2D object at 0x000001E38E652490>]
[14.79137904]
```



9. Write a program to demonstrate the working of Random Forest classifier. Useappropriate dataset for Random Forest Classifier.

Introduction

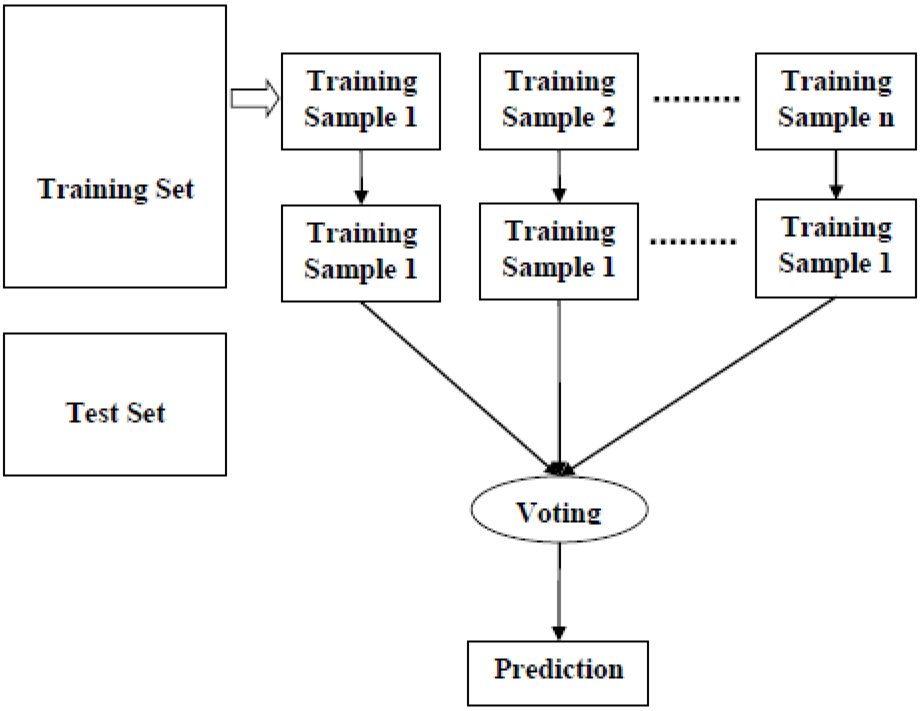
Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Working of Random Forest Algorithm

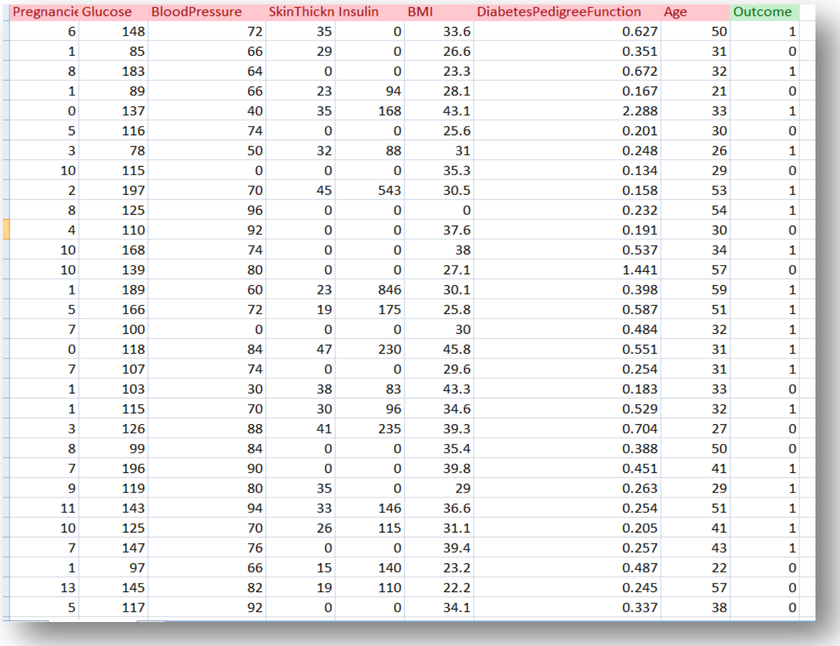
We can understand the working of Random Forest algorithm with the help of following steps −

- **Step 1** − First, start with the selection of random samples from a given dataset.
- **Step 2** − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- **Step 3** − In this step, voting will be performed for every predicted result.
- **Step 4** − At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working −



Data Set : diabetes2.csv



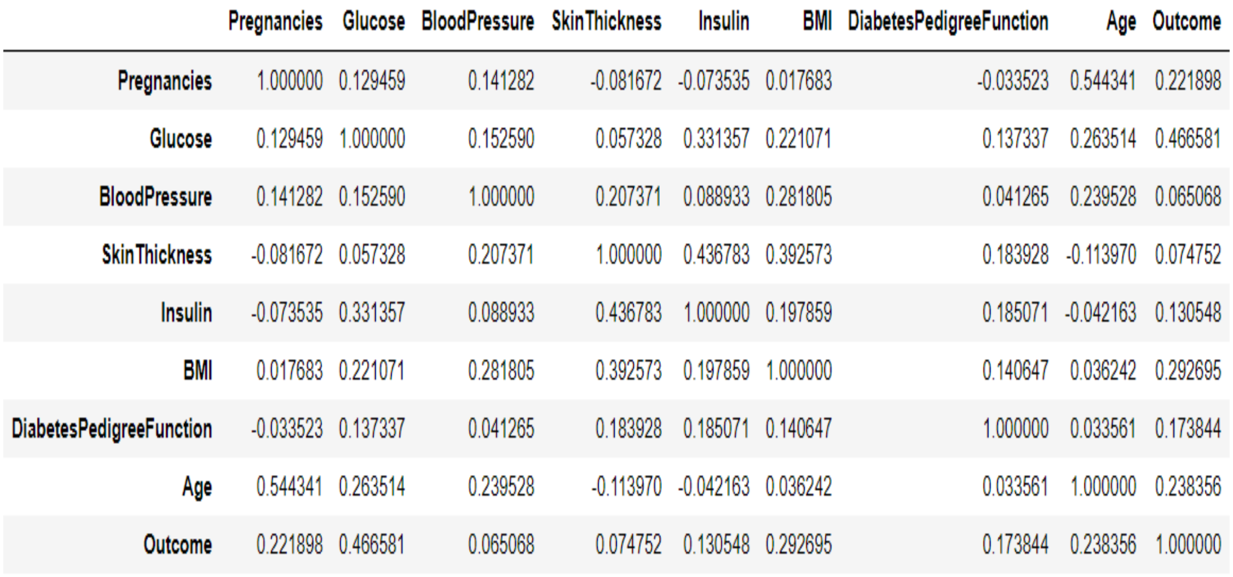
Pregnancie	Glucose	BloodPressure	SkinThickn	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1
10	125	70	26	115	31.1	0.205	41	1
7	147	76	0	0	39.4	0.257	43	1
1	97	66	15	140	23.2	0.487	22	0
13	145	82	19	110	22.2	0.245	57	0
5	117	92	0	0	34.1	0.337	38	0

< program >

```
import pandas as pd

from     sklearn.ensemble     import
RandomForestClassifier          from
sklearn.model_selection      import
train_test_split                   from
sklearn.metrics                   import
confusion_matrix


# Load the dataset
df = pd.read_csv("diabetes2.csv")
# Split the dataset into features and target
X                =
df.drop("Out
come",
axis=1)  y   =
df["Outc-
ome"]
```

#Correlation

```
data.corr()
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

# Split the dataset into training and testing data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=0)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

O/P;

```
(576, 8)
(192, 8)
(576,)
(192,)
```

# Create the random forest classifier and fit the model using the training data

```
classifier =
RandomForestClassifier(n_estimators=100,
  random_state=0)classifier.fit(X_train, y_train)
```

O/P:

```
RandomForestClassifier()
```

```
  # Make predictions on the test data

  y_pred = classifier.predict(X_test)
print(y_pred)
```

```
[1 1 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0
 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1
 0 0 0 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1
 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1
 1 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0
 0 0 0 1 0 0 0 1 0]
```

```
  #       Print       the

  confusion     matrix

  cm                  =

  confusion_matrix(y

  _test,         y_pred)

  print("Confusion

  Matrix:")


  cm
```

O/P:

```
Confusion Matrix:
Out[22]:
array([[115,  12],
       [ 31,  34]], dtype=int64)
```

```
# Print the classification report

from  sklearn.metrics  import

classification_report   res    =

classification_report(y_test,

y_pred)

print("\nClassification

Report:\n", res)
```

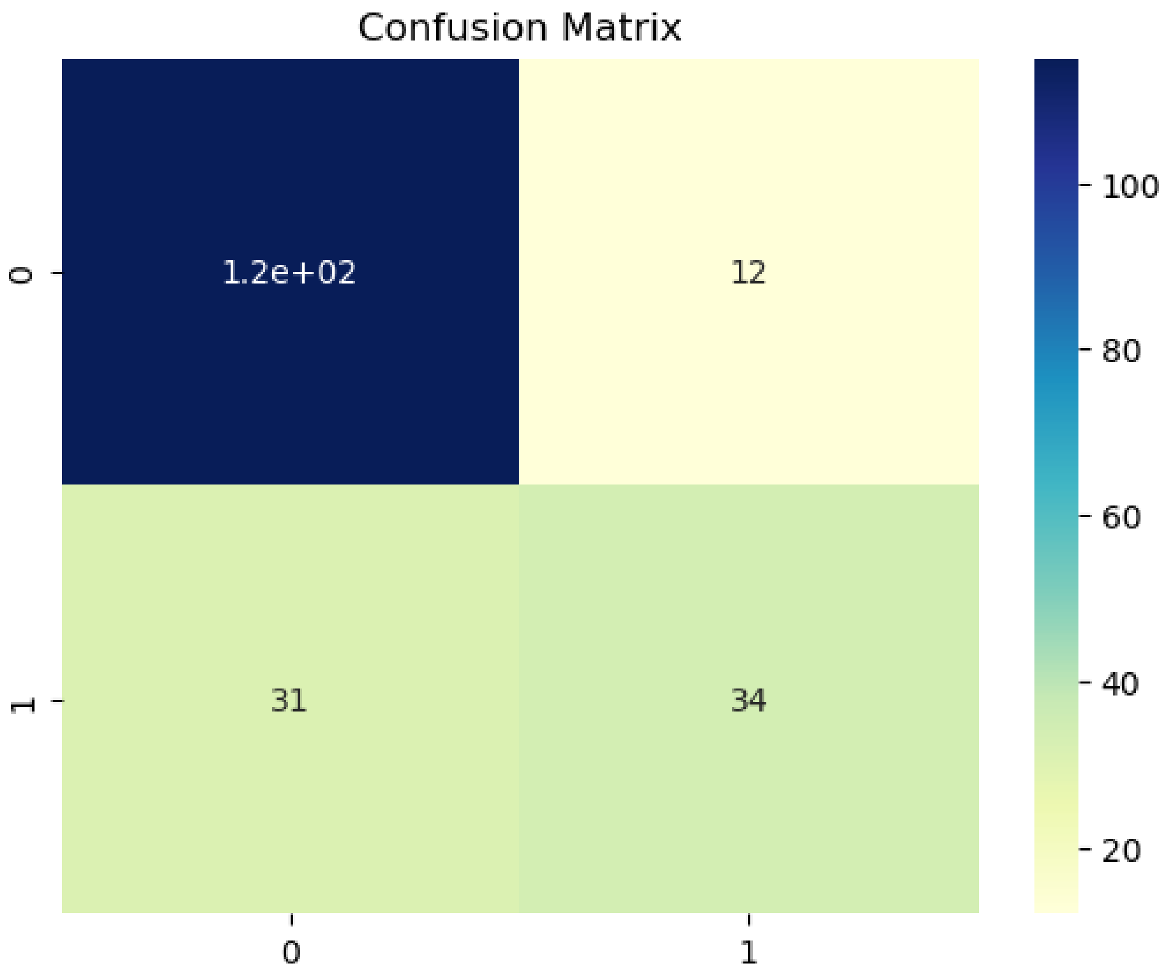
O/P:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.91      0.84       127
           1       0.74      0.52      0.61        65

    accuracy                           0.78       192
   macro avg       0.76      0.71      0.73       192
weighted avg       0.77      0.78      0.76       192
```

# Generate the confusion matrix
# Create the heatmap using seaborn

```
import seaborn as sns

  import matplotlib.pyplot as plt


  from sklearn.metrics import confusion_matrix

  sns.heatmap(cm,annot=True,cmap="YlGnBu")
```

```
plt.title("Confusion Matrix")
```



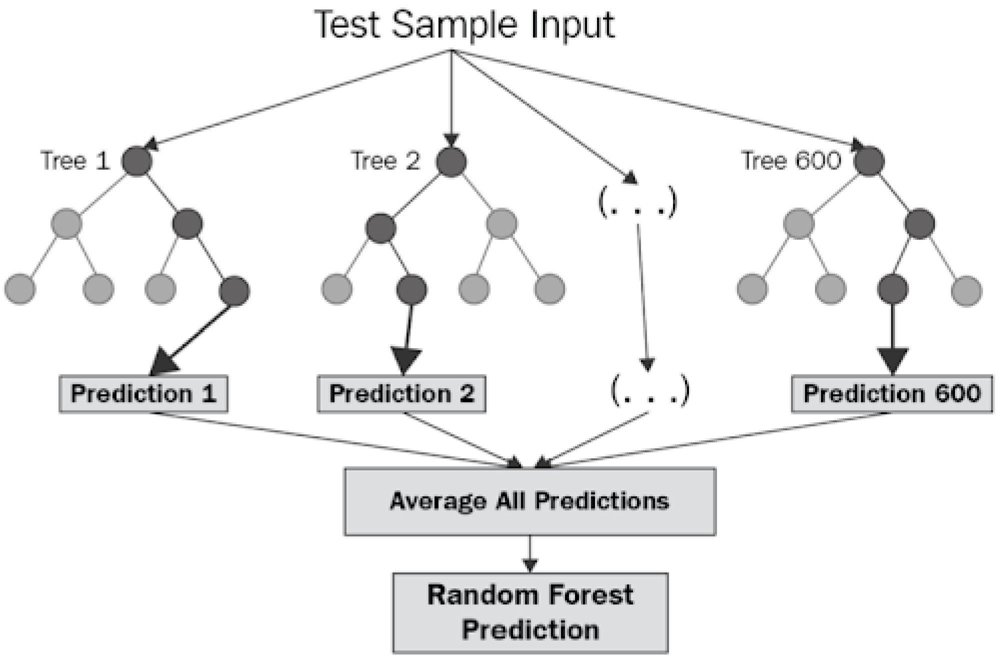
```
from sklearn.metrics import precision_score,recall_score, f1_score, accuracy_score,confusion_matrix
print("Accuracy:", accuracy_score(y_test,y_pred))

Accuracy: 0.7760416666666666

print("Precision:", precision_score(y_test,y_pred,average="weighted"))

Precision: 0.7712381501886044


print('Recall:', recall_score(y_test,y_pred,average="weighted"))

Recall: 0.7760416666666666
```

10. Write a program to demonstrate the working of Random Forest Regressor. Use appropriate dataset for Random Forest Regressor.

Random Forest Regressor:

Random forest regression is a supervised learning algorithm that uses an ensemble learning method for regression.

Random forest is a bagging technique and not a boosting technique. The trees in random forests run in parallel, meaning there is no interaction between these trees while building the trees.



Algorithm:

- Design a specific question or data and get the source to determine the required data.
- Make sure the data is in an accessible format else convert it to the required format.

- Specify all noticeable anomalies and missing data points that may be required to achieve the required data.
- Create a machine-learning model.
- Set the baseline model that you want to achieve
- Train the data machine learning model.
- Provide an insight into the model with test data
- Now compare the performance metrics of both the test data and the predicted data from the model.
- If it doesn't satisfy your expectations, you can try improving your model accordingly or dating your data, or using another data modeling technique.
- At this stage, you interpret the data you have gained and report accordingly.

Important Hyperparameters in Random Forest

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

Hyperparameters to Increase the Predictive Power

n_estimators: Number of trees the algorithm builds before averaging the predictions.

max_features: Maximum number of features random forest considers splitting a node.

mini_sample_leaf: Determines the minimum number of leaves required to split an internal node.

criterion: How to split the node in each tree? (Entropy/Gini impurity/Log Loss)

max_leaf_nodes: Maximum leaf nodes in each tree

Hyperparameters to Increase the Speed

n_jobs: it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor, but if the value is -1, there is no limit.

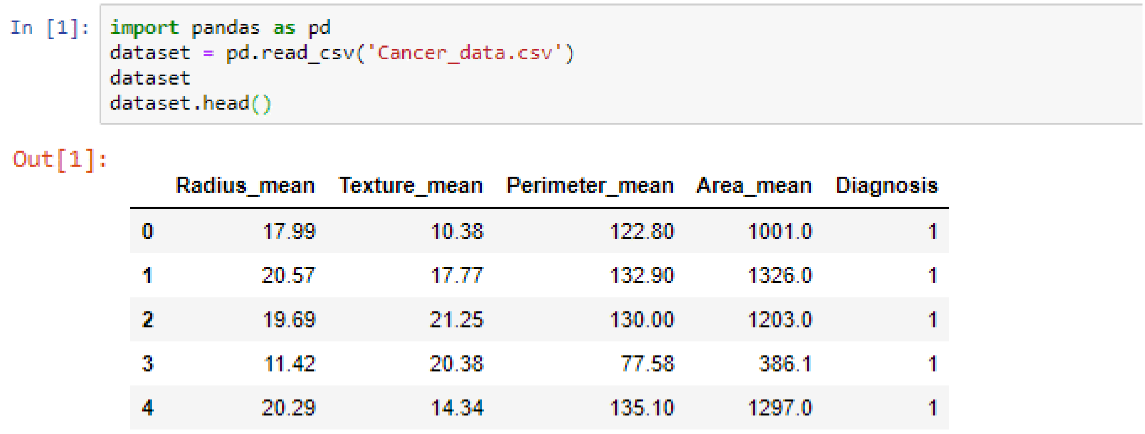
random_state: controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and has been given the same hyperparameters and training data.

oob_score: *OOB* means out of the bag. It is a random forest cross-validation method. In this, one-third of the sample is not used to train the data; instead used to evaluate its performance. These samples are called out-of-bag samples.

Python code:

Step 1: Load Pandas library and the dataset using Pandas

Step 1: Load Pandas library and the dataset using Pandas

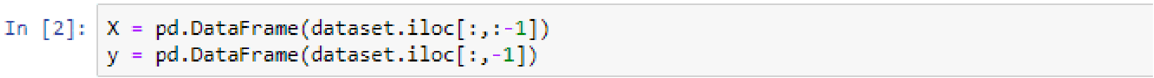


```
In [1]: import pandas as pd
        dataset = pd.read_csv('Cancer_data.csv')
        dataset
        dataset.head()
```

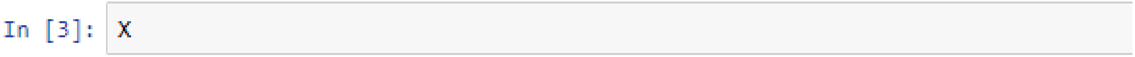
Out[1]:

	Radius_mean	Texture_mean	Perimeter_mean	Area_mean	Diagnosis
0	17.99	10.38	122.80	1001.0	1
1	20.57	17.77	132.90	1326.0	1
2	19.69	21.25	130.00	1203.0	1
3	11.42	20.38	77.58	386.1	1
4	20.29	14.34	135.10	1297.0	1

Step 2: Define the features and the target

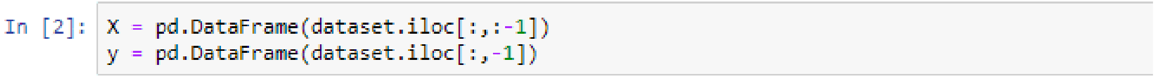


```
In [2]: X = pd.DataFrame(dataset.iloc[:,:-1])
        y = pd.DataFrame(dataset.iloc[:,-1])
```

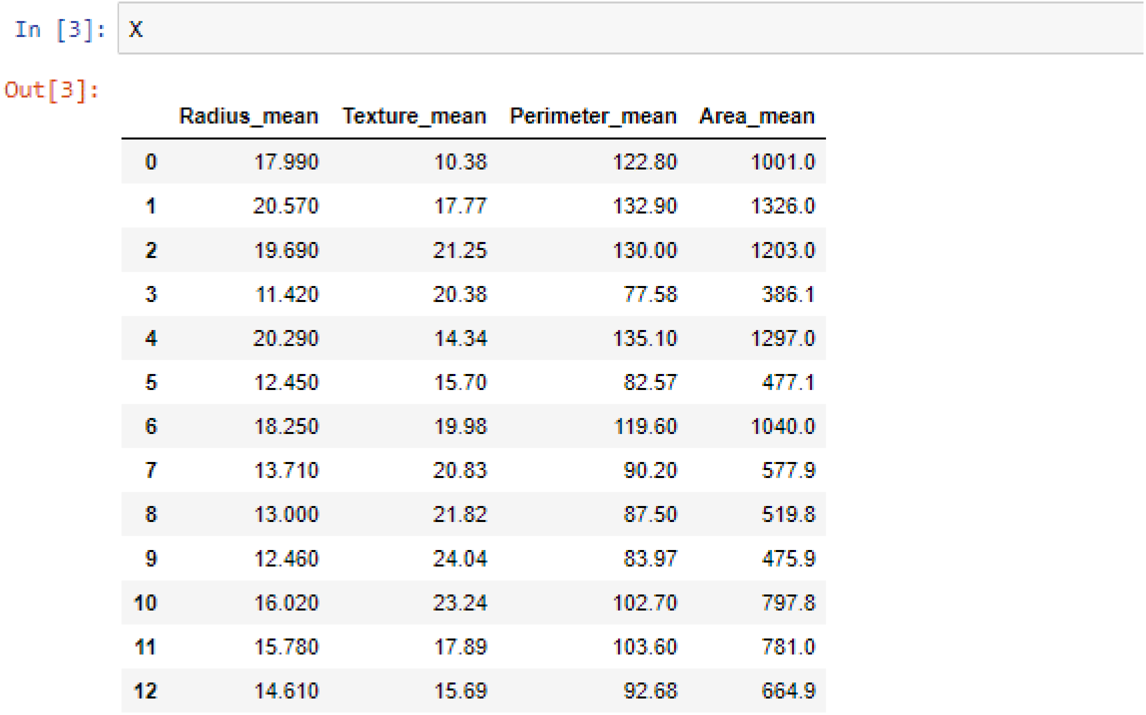


```
In [3]: X
```

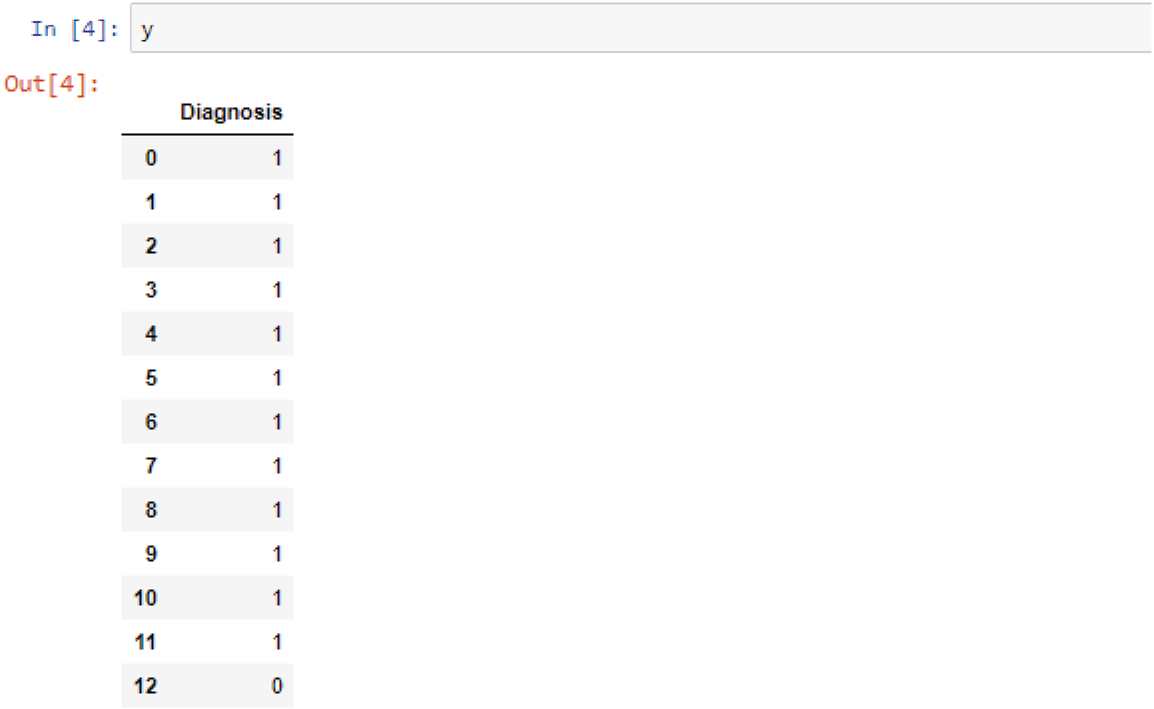
Step 2: Define the features and the target



```
In [2]: X = pd.DataFrame(dataset.iloc[:,:-1])
        y = pd.DataFrame(dataset.iloc[:,-1])
```

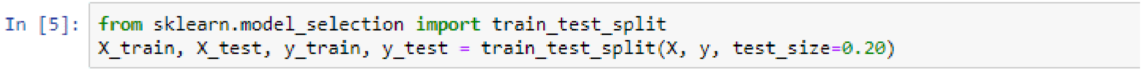


```
In [3]: X

Out[3]:
        Radius_mean  Texture_mean  Perimeter_mean  Area_mean
    0        17.990         10.38          122.80     1001.0
    1        20.570         17.77          132.90     1326.0
    2        19.690         21.25          130.00     1203.0
    3        11.420         20.38           77.58      386.1
    4        20.290         14.34          135.10     1297.0
    5        12.450         15.70           82.57      477.1
    6        18.250         19.98          119.60     1040.0
    7        13.710         20.83           90.20      577.9
    8        13.000         21.82           87.50      519.8
    9        12.460         24.04           83.97      475.9
   10        16.020         23.24          102.70      797.8
   11        15.780         17.89          103.60      781.0
   12        14.610         15.69           92.68      664.9
```



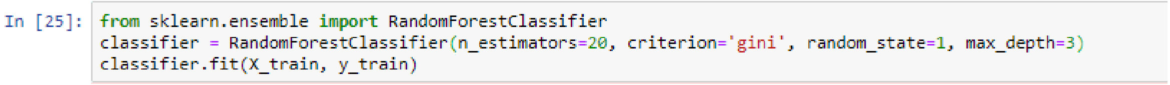
```
In [4]: y

Out[4]:
        Diagnosis
    0           1
    1           1
    2           1
    3           1
    4           1
    5           1
    6           1
    7           1
    8           1
    9           1
   10           1
   11           1
   12           0
```

Step 3: Split the dataset into train and test sklearn



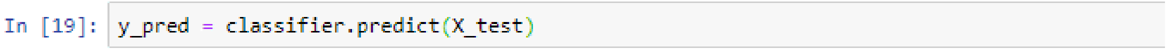
```
In [5]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

Step 4: Import the random forest classifier function from sklearn ensemble module. Build the random forest classifier model with the help of the random forest classifier function



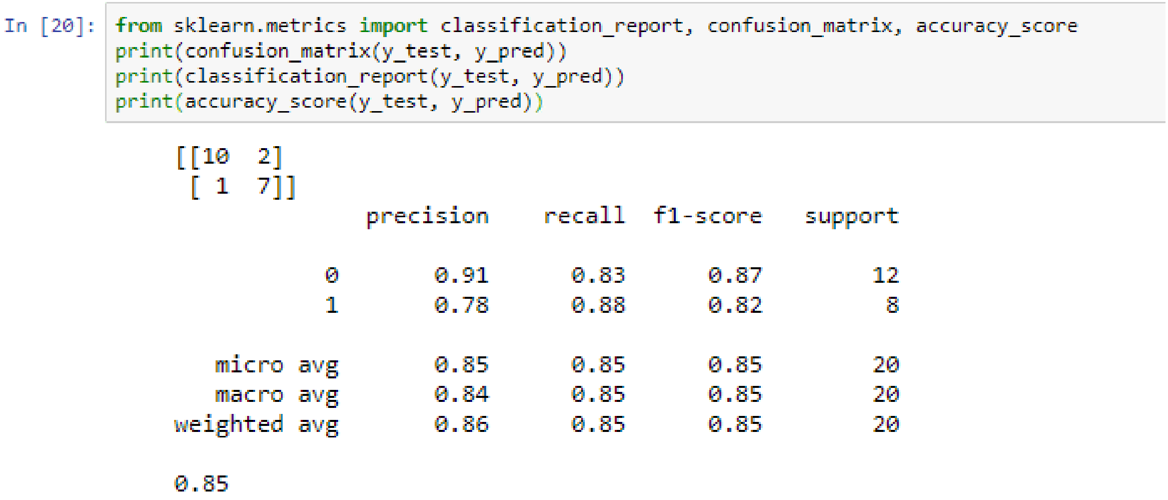
```
In [25]: from sklearn.ensemble import RandomForestClassifier
         classifier = RandomForestClassifier(n_estimators=20, criterion='gini', random_state=1, max_depth=3)
         classifier.fit(X_train, y_train)
```

Step 5: Predict values using the random forest classifier model



```
In [19]: y_pred = classifier.predict(X_test)
```

Step 6: Evaluate the random forest classifier model



```
In [20]: from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
         print(confusion_matrix(y_test, y_pred))
         print(classification_report(y_test, y_pred))
         print(accuracy_score(y_test, y_pred))

[[10  2]
 [ 1  7]]
              precision    recall  f1-score   support

           0       0.91      0.83      0.87        12
           1       0.78      0.88      0.82         8

   micro avg       0.85      0.85      0.85        20
   macro avg       0.84      0.85      0.85        20
weighted avg       0.86      0.85      0.85        20

0.85
```

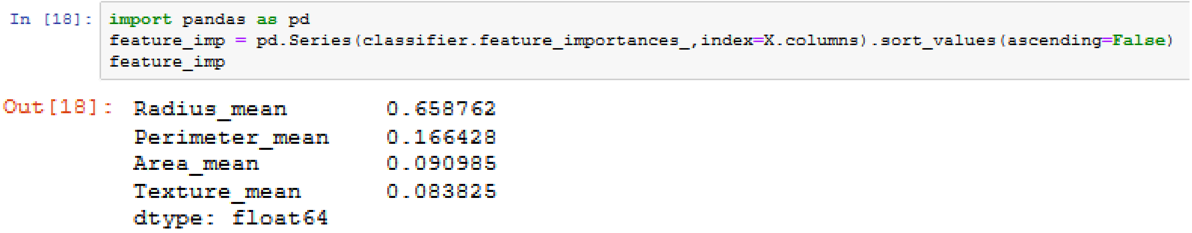
Feature Selection in Random Forest Algorithm Model

With the help of Scikit-Learn, we can select important features to build the random forest algorithm model in order to avoid the overfitting issue. There are two ways to do this:

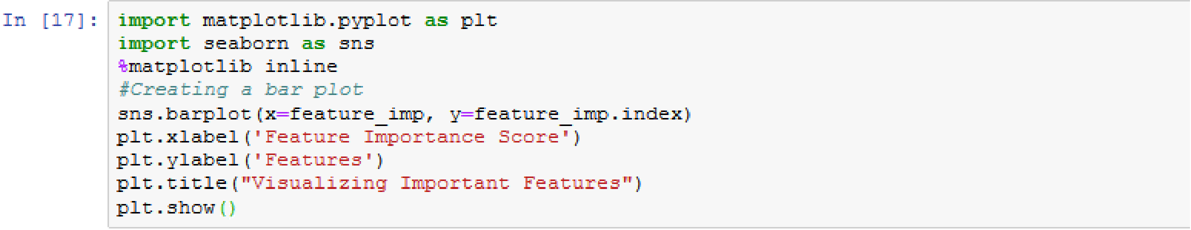
- Visualize which feature is not adding any value to the model
- Take help of the built-in function **SelectFromModel**, which allows us to add a threshold value to neglect features below that threshold value.

Let us see if selecting features make any difference in the accuracy score of the model.

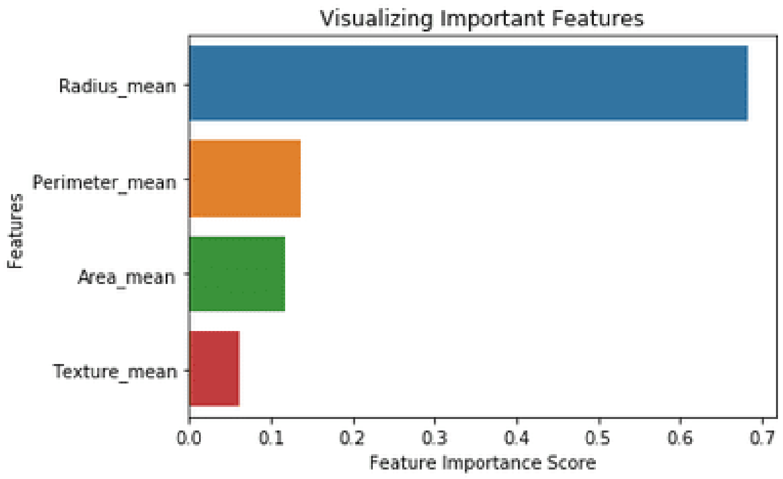
Step 7: Let us find out important features and visualize them using Seaborn



```
In [18]: import pandas as pd
         feature_imp = pd.Series(classifier.feature_importances_,index=X.columns).sort_values(ascending=False)
         feature_imp

Out[18]: Radius_mean       0.658762
         Perimeter_mean    0.166428
         Area_mean         0.090985
         Texture_mean      0.083825
         dtype: float64
```

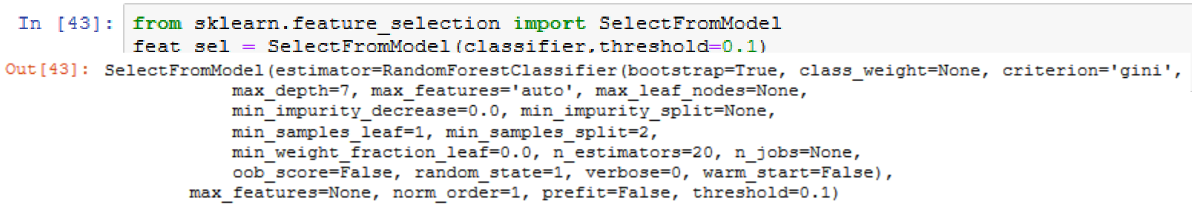


```
In [17]: import matplotlib.pyplot as plt
         import seaborn as sns
         %matplotlib inline
         #Creating a bar plot
         sns.barplot(x=feature_imp, y=feature_imp.index)
         plt.xlabel('Feature Importance Score')
         plt.ylabel('Features')
         plt.title("Visualizing Important Features")
         plt.show()
```



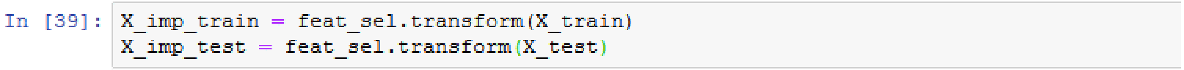
Now let us see how the ‘SelectFromModel’ function helps in building a random forest classifier model with important features.

Step 8: Import the SelectFromModel function. We will pass the classifier object we’ve created above. Also, we will add a threshold value of 0.1



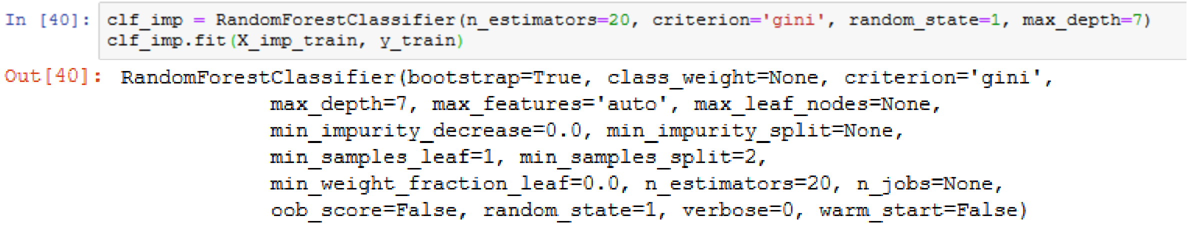
```
In [43]: from sklearn.feature_selection import SelectFromModel
         feat_sel = SelectFromModel(classifier,threshold=0.1)
Out[43]: SelectFromModel(estimator=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=7, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=20, n_jobs=None,
                     oob_score=False, random_state=1, verbose=0, warm_start=False),
                 max_features=None, norm_order=1, prefit=False, threshold=0.1)
```

Step 9: With the help of the ‘transform’ method, we will pick the important features and store them in new train and test objects



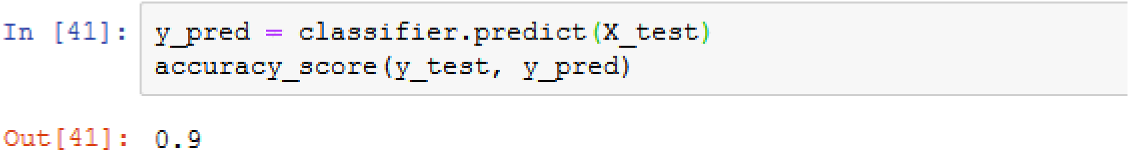
```
In [39]: X_imp_train = feat_sel.transform(X_train)
         X_imp_test = feat_sel.transform(X_test)
```

Step 10: Let us now build a new random forest classifier model (so that we can compare the results of this model with the old one)



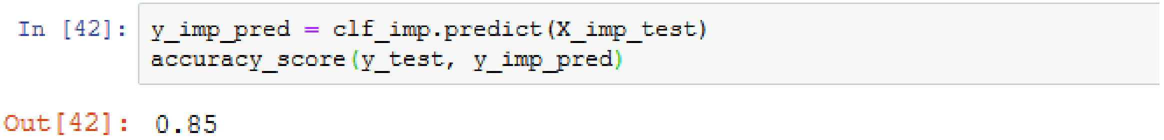
```
In [40]: clf_imp = RandomForestClassifier(n_estimators=20, criterion='gini', random_state=1, max_depth=7)
         clf_imp.fit(X_imp_train, y_train)
Out[40]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=7, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=20, n_jobs=None,
                     oob_score=False, random_state=1, verbose=0, warm_start=False)
```

Step 11: Let us see the accuracy result of the old model



```
In [41]: y_pred = classifier.predict(X_test)
         accuracy_score(y_test, y_pred)

Out[41]: 0.9
```

Step 12: Let us see the accuracy result of the new model after feature selection



```
In [42]: y_imp_pred = clf_imp.predict(X_imp_test)
         accuracy_score(y_test, y_imp_pred)

Out[42]: 0.85
```

Note: After the feature selection process, the accuracy score is decreased. But, we have successfully picked out the important features at a small cost of accuracy.

Also, automatic feature selection reduces the complexity of the model but does not necessarily increase the accuracy. In order to get the desired accuracy, we have to perform the feature selection process manually.

Step 13: To find Confusion Matrix:

```
y=confusion_matrix(y_test,y_pred)

y
```

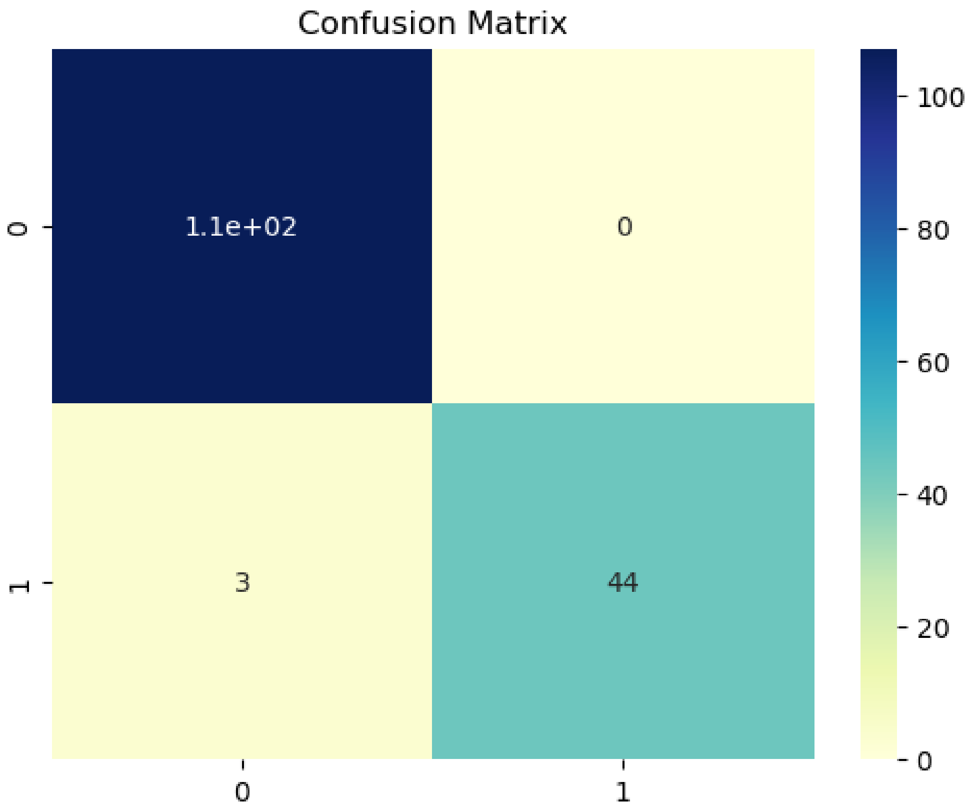
Out[71]:

```
array([[107,   0],
       [  3,  44]], dtype=int64)
```

```
import seaborn as sns

sns.heatmap(y,annot=True,cmap="YlGnBu")

plt.title("Confusion Matrix")
```

Out[72]:

```
Text(0.5, 1.0, 'Confusion Matrix')
```


Confusion Matrix
1.1e+02
0
3
44
0
1
0
1
100
80
60
40
20
0

Artificial Intelligence

1) AIM: Implementation of DFS for water jug problem:

Explanation:

Water Jug Problem is one of the most important problems to solve in Java. The water jug problem is a problem where we have two jugs, "**i**" liter jug and **"j"** liter jug **(0 < i < j)**. Both jugs will initially be empty, and they don't have marking to measure small quantities. Now, we need to measure d liters of water by using these two jugs where d < j. We use the following three operations to measure small quantities by using the two jars:

1. Empty a Jug.
2. Fill a Jug
3. We pour the water of one jug into another one until one of them is either full or empty.

In Java, we implement the logic for getting the minimum number of operations required to measure the d liter quantity of water.

There are various ways to solve water jug problems in Java, including GCD, BFS, and DP. In this section, we implement the logic for solving the Water Jug problem by using GCD.

Example: Water Jug Problem

Consider the following problem:

S.No.	Initial State	Condition	Final state	Description of action taken
1.	(x,y)	If x<4	(4,y)	Fill the 4 gallon jug completely

2.	(x,y)	if y<3	(x,3)	Fill the 3 gallon jug completely
3.	(x,y)	If x>0	(x-d,y)	Pour some part from the 4 gallon jug
4.	(x,y)	If y>0	(x,y-d)	Pour some part from the 3 gallon jug
5.	(x,y)	If x>0	(0,y)	Empty the 4 gallon jug
6.	(x,y)	If y>0	(x,0)	Empty the 3 gallon jug
7.	(x,y)	If (x+y)<7	(4, y-[4-x])	Pour some water from the 3 gallon jug to fill the four gallon jug
8.	(x,y)	If (x+y)<7	(x-[3-y],y)	Pour some water from the 4 gallon jug to fill the 3 gallon jug.

9.	(x,y)	If (x+y)<4	(x+y,0)	Pour all water from 3 gallon jug to the 4 gallon jug
10.	(x,y)	if (x+y)<3	(0, x+y)	Pour all water from the 4 gallon jug to the 3 gallon jug

A Water Jug Problem: You are given two jugs, a 4-gallon one and a 3-gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

State Representation and Initial State – we will represent a state of the problem as a

tuple (x, y) where x represents the amount of water in the 4-gallon jug and y represents the amount of water in the 3-gallon jug. Note $0 \le x \le 4$, and $0 \le y \le 3$.

Our initial state: (0,0)

Goal Predicate – state = (2,y) where $0 \le y \le 3$.

Production rules for solving the water jug problem

Here, let ***x*** denote the 4-gallon jug and ***y*** denote the 3-gallon jug.

The listed production rules contain all the actions that could be performed by the agent in transferring the contents of jugs. But, to solve the water jug problem in a minimum number of moves, following set of rules in the given sequence should be performed:

Solution of water jug problem according to the production rules

S.No.	4 gallon jug contents	3 gallon jug contents	Rule followed
1.	0 gallon	0 gallon	Initial state
2.	0 gallon	3 gallons	Rule no.2
3.	3 gallons	0 gallon	Rule no. 9
4.	3 gallons	3 gallons	Rule no. 2
5.	4 gallons	2 gallons	Rule no. 7
6.	0 gallon	2 gallons	Rule no. 5
7.	2 gallons	0 gallon	Rule no. 9

On reaching the 7th attempt, we reach a state which is our goal state. Therefore, at this state, our problem is solved

Implementation of water jug problem Using Python:

```
print("Rule 1:Fill x\n Rule 2:Fill y\n Rule 3:Empty x\n Rule 4:Empty y\n Rule 5:From y to x\n Rule 6:From x to y\n Rule 7:From y to x complete\n Rule 8:From x to y complete\n")
cap_x = int(input("Enter the jug 1 capacity: "))
cap_y = int(input("Enter the jug 2 capacity: "))
req_lis = list(map(str,input("Enter the required amount of water and in the jug you needed with space seperate: ").split()))
```

```
req_amount = int(req_lis[0])
req_jug = req_lis[1]
x=y=0
while(True):
    rule = int(input("Enter the rule: "))
    if rule==1:
        if x<cap_x:
            x = cap_x
    if rule==2:
        if y<cap_y:
            y = cap_y
    if rule==3:
        if x>0:
            x = 0
    if rule==4:
        if y>0:
            y = 0
    if rule==5:
        if 0<x+y>=cap_x and y>0:
            x,y = cap_x,y-(cap_x-x)
    if rule==6:
        if 0<x+y>=cap_y and x>0:
            x,y = x-(cap_y-y),cap_y
    if rule==7:
        if 0<x+y<=cap_x and y>=0:
            x = x+y
            y = 0
    if rule==8:
        if 0<x+y<=cap_y and x>=0:
            y = x+y
            x = 0
    print("x :",x)
    print("y :",y)
    if req_jug=='x':
        if req_amount==x:
            print("Goal reached")
            break
    elif req_jug=='y':
        if req_amount==y:
```

```
        print("Goal reached")
        break
```

output:

```
Rule 1:Fill x

Rule 2:Fill y

Rule 3:Empty x

Rule 4:Empty y

Rule 5:From y to x

Rule 6:From x to y

Rule 7:From y to x complete

Rule 8:From x to y complete


Enter the jug 1 capacity: 4

Enter the jug 2 capacity: 3

Enter the required amount of water and in the jug you needed with space seperate: 2 x

Enter the rule: 1

x : 4

y : 0

Enter the rule: 6

x : 1

y : 3

Enter the rule: 4

x : 1

y : 0

Enter the rule: 8

x : 0

y : 1
```

```
Enter the rule: 1
x : 4
y : 1
Enter the rule: 6
x : 2
y : 3
Goal reached
```

Implementation of water jug problem Using Java:

```
import java.util.Scanner;
public class Main
{
public static void main(String[] args) {
System.out.println("WATER JUG PROBLEM");
Scanner res=new Scanner(System.in);
  System.out.println("ENTER CAPACITY OF JUG-1 :");
  int x=res.nextInt();
  System.out.println("ENTER CAPACITY OF JUG-2 :");
  int y=res.nextInt();
  System.out.println("ENTER THE GOAL STATE :");
  int a=res.nextInt();
  do{
    System.out.println("ENTER rule num :");
    int rule=res.nextInt();
    if (rule==1){
       if (x<4)
            x=4;
    }
```

```
       else if (rule==2)
          {
             if (y<3)
               y=3;
          }
       else if (rule==3)
          {
              if (x>0)
                x=0;
          }
       else if (rule==4)
        {
           if (y>0)
           y=0;
        }
      else if (rule==5)
        {
           if (x+y>=4 && y>0)
           y=y-(4-x);x=4;
        }



else if (rule==6)
       {
          if (x+y>=3 && x>0)
           x=x-(3-y);y=3;
       }
      else if (rule==7)
        {
            if (x+y<=4 && y>=0)
            x=x+y;y=0;
        }
    else if (rule==8)
        {
             if (x+y<=4 && y>=0)
               y=x+y;x=0;
        }
    //if (x==a || y==a)
        //System.out.println("goal reached");
        //break;
    System.out.println("x= "+x);
    System.out.println("y= "+y);

   }while(x!=a && y!=a);
   System.out.println("GOAL REACHED");
}
}
```

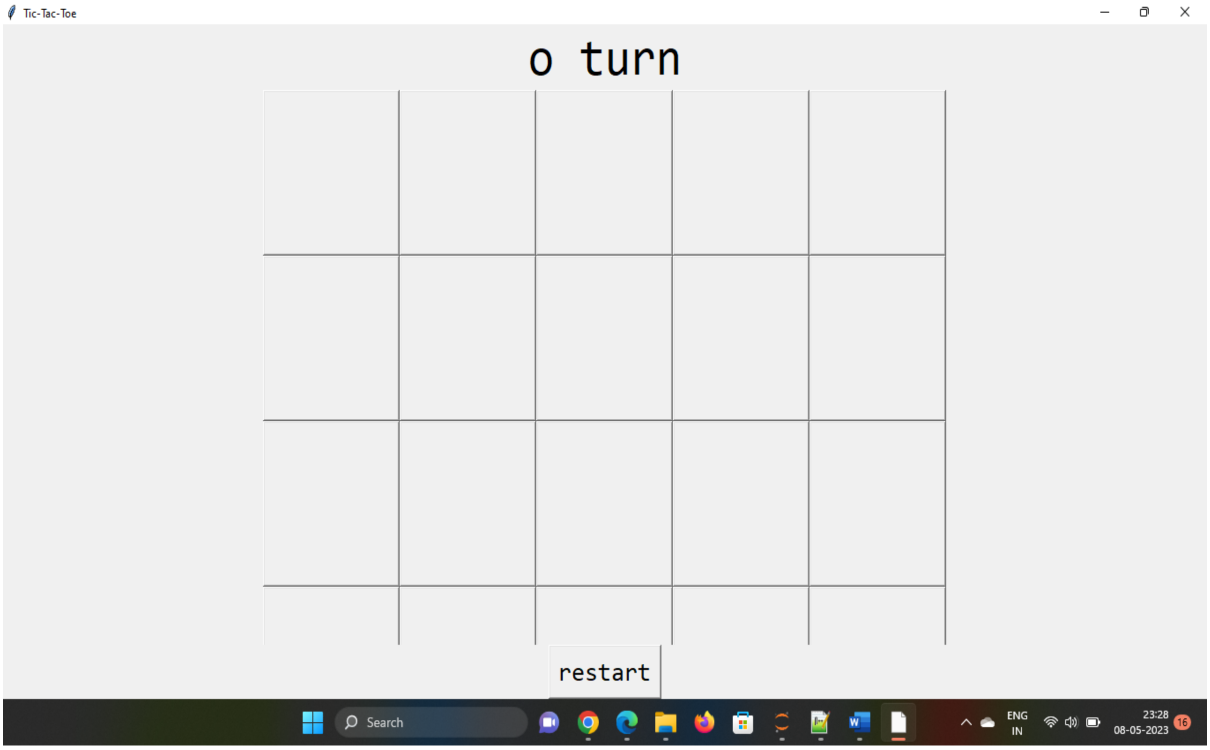
2) Aim: Implement and demonstrate the Tic-Tac-Toe problem in python code.

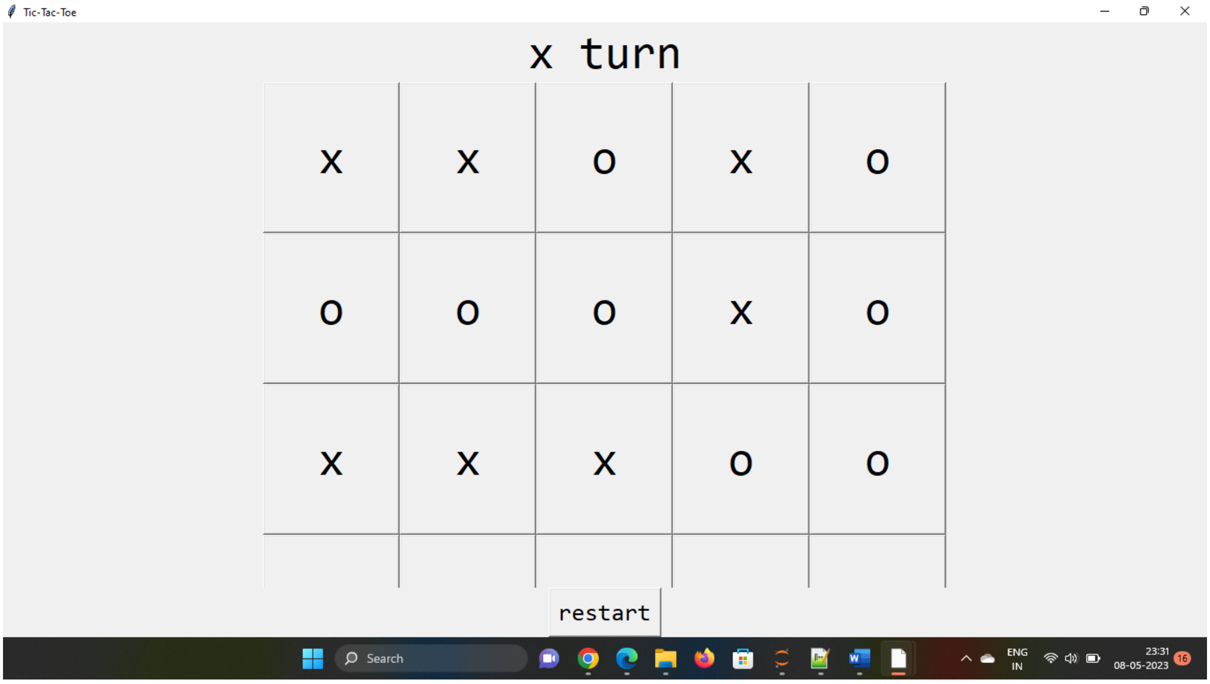
Explanation:

There will be two players in a game. Two signs represent each player. The general signs used in the game are **X** and **O**

```
from tkinter import *
import random

def next_turn(row, column):

    global player

    if buttons[row][column]['text'] == "" and check_winner() is False:

        if player == players[0]:

            buttons[row][column]['text'] = player

            if check_winner() is False:
                player = players[1]
                label.config(text=(players[1]+" turn"))

            elif check_winner() is True:
                label.config(text=(players[0]+" wins"))

            elif check_winner() == "Tie":
                label.config(text="Tie!")

        else:

            buttons[row][column]['text'] = player

            if check_winner() is False:
                player = players[0]
                label.config(text=(players[0]+" turn"))

            elif check_winner() is True:
                label.config(text=(players[1]+" wins"))

            elif check_winner() == "Tie":
                label.config(text="Tie!")

def check_winner():

    for row in range(5):
        if buttons[row][0]['text'] == buttons[row][1]['text'] == buttons[row][2]['text'] == buttons[row][3]['text'] == buttons[row][4]['text'] != "":
            buttons[row][0].config(bg="green")
            buttons[row][1].config(bg="green")
            buttons[row][2].config(bg="green")
            buttons[row][3].config(bg="green")
            buttons[row][4].config(bg="green")
            return True

    for column in range(5):
        if buttons[0][column]['text'] == buttons[1][column]['text'] ==
buttons[2][column]['text'] == buttons[3][column]['text'] ==
buttons[4][column]['text']!= "":
            buttons[0][column].config(bg="green")
            buttons[1][column].config(bg="green")
            buttons[2][column].config(bg="green")
            buttons[3][column].config(bg="green")
            buttons[4][column].config(bg="green")
            return True

    if buttons[0][0]['text'] == buttons[1][1]['text'] == buttons[2][2]['text'] ==
buttons[3][3]['text'] == buttons[4][4]['text']!= "":
        buttons[0][0].config(bg="green")
        buttons[1][1].config(bg="green")
        buttons[2][2].config(bg="green")
        buttons[3][3].config(bg="green")
        buttons[4][4].config(bg="green")
        return True

    elif buttons[0][4]['text'] == buttons[1][3]['text'] == buttons[2][2]['text'] ==
buttons[3][1]['text'] == buttons[4][0]['text']!= "":
        buttons[0][4].config(bg="green")
        buttons[1][3].config(bg="green")
        buttons[2][2].config(bg="green")
        buttons[3][1].config(bg="green")
        buttons[4][0].config(bg="green")
        return True

    elif empty_spaces() is False:

        for row in range(5):
            for column in range(5):
                buttons[row][column].config(bg="yellow")
        return "Tie"

    else:
        return False


def empty_spaces():

    spaces = 25

    for row in range(5):
        for column in range(5):
            if buttons[row][column]['text'] != "":
                spaces -= 1

    if spaces == 0:
        return False
    else:
        return True

def new_game():

    global player

    player = random.choice(players)

    label.config(text=player+" turn")

    for row in range(5):
        for column in range(5):
            buttons[row][column].config(text="",bg="#F0F0F0")


window = Tk()
window.title("Tic-Tac-Toe")
players = ["x","o"]
player = random.choice(players)
buttons = [[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]]

label = Label(text=player + " turn", font=('consolas',40))
label.pack(side="top")
```

```
reset_button = Button(text="restart", font=('consolas',20),
command=new_game)
reset_button.pack(side="bottom")

frame = Frame(window)
frame.pack()

for row in range(5):
    for column in range(5):
        buttons[row][column] = Button(frame, text="",font=('consolas',40),
width=5, height=2,
                                      command= lambda row=row, column=column:
next_turn(row,column))
        buttons[row][column].grid(row=row,column=column)

window.mainloop()
```





x turn

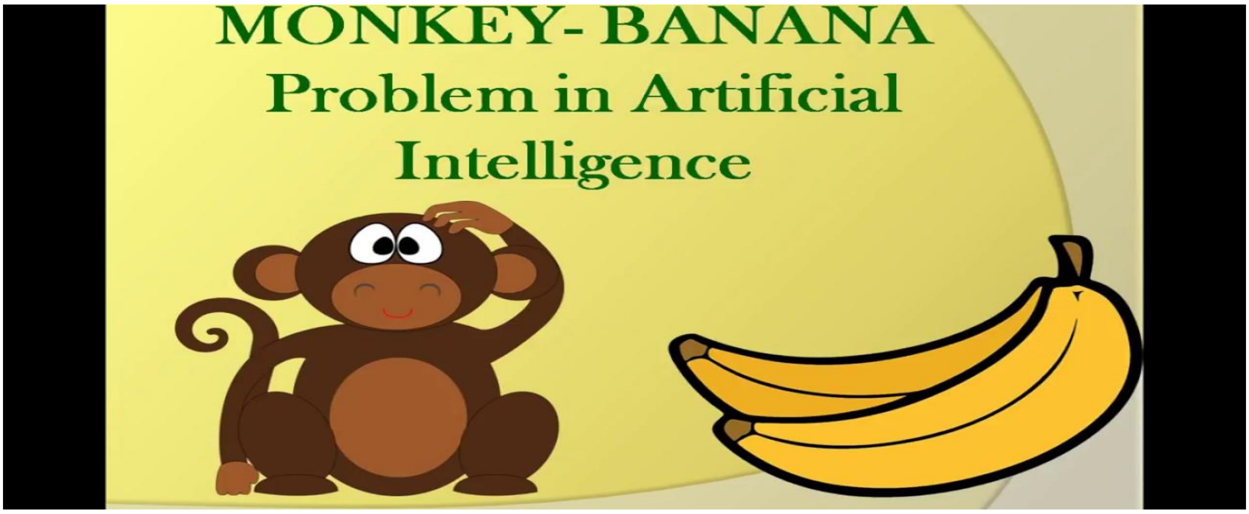
X	X	O	X	O
O	O	O	X	O
X	X	X	O	O

3) Aim:-Implementation of Monkey Banana Problem using LISP/PROLOG

Problem Statement:-

Suppose the problem is as given below –

- A hungry monkey is in a room, and he is near the door.
- The monkey is on the floor.
- Bananas have been hung from the center of the ceiling of the room.
- There is a block (or chair) present in the room near the window.
- The monkey wants the banana, but cannot reach it.



So how can the monkey get the bananas?

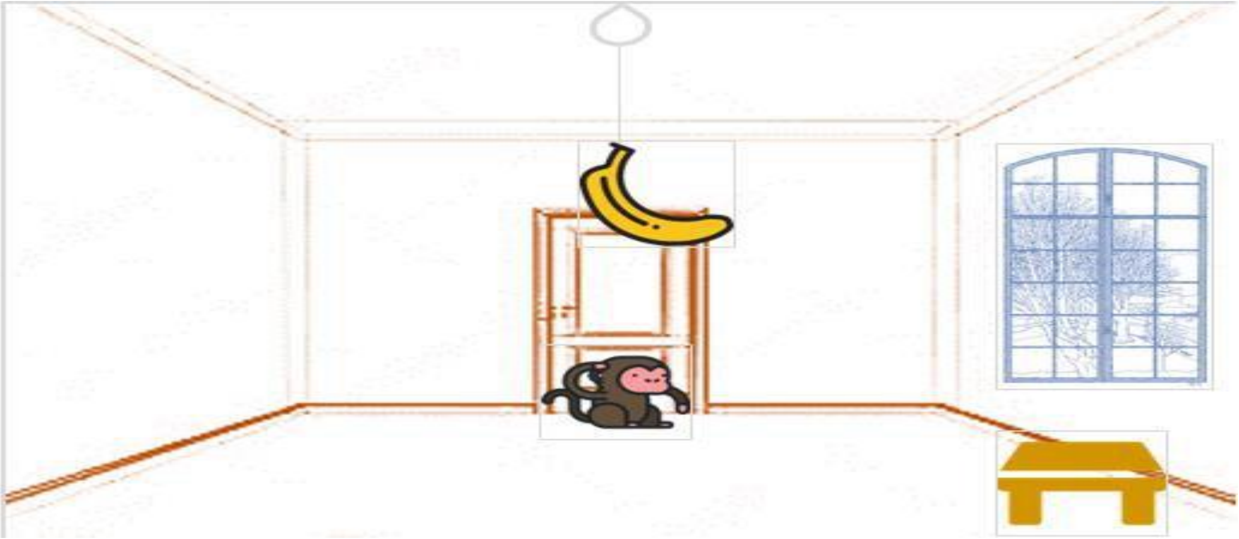
So if the monkey is clever enough, he can come to the block, drag the block to the center, climb on it, and get the banana. Below are few observations in this case −

- Monkey can reach the block, if both of them are at the same level. From the above image, we can see that both the monkey and the block are on the floor.
- If the block position is not at the center, then monkey can drag it to the center.
- If monkey and the block both are on the floor, and block is at the center, then the monkey can climb up on the block. So the vertical position of the monkey will be changed.
- When the monkey is on the block, and block is at the center, then the monkey can get the bananas.

Now, let us see how we can solve this using Prolog. We will create some predicates as follows −

We have some predicates that will move from one state to another state, by performing action.

- When the block is at the middle, and monkey is on top of the block, and monkey does not have the banana (i.e. ***has not*** state), then using the ***grasp*** action, it will change from ***has not*** state to ***have*** state.
- From the floor, it can move to the top of the block (i.e. ***on top*** state), by performing the action ***climb***.
- The **push** or **drag** operation moves the block from one place to another.



Monkey banana using prolog:-

```
on(floor,monkey).

on(floor,box).

in(room,monkey).

in(room,box).

at(ceiling,banana).

strong(monkey).

grasp(monkey).

climb(monkey,box).

push(monkey,box):-

        strong(monkey).

under(banana,box):-

        push(monkey,box).

canreach(banana,monkey):-

        at(floor,banana);

      at(ceiling,banana),

      under(banana,box),

      climb(monkey,box).

canget(banana,monkey):-

        canreach(banana,monkey),

      grasp(monkey).
```

o/p:-

```
?-['E:/monkey.pl].

True

?- canget(banana,monkey):-

True.
```