Name: Sourav Patra

Roll No: 001811001044

Department: Information Technology

Year: 4th

Machine learning Assignment I

IRIS PLANT DATASET

2. Decision Tree classifier

First we will import some packages like numpy,pandas and matplotlib for variety of mathematical operation.Then we will read the iris data which I have downloaded from below website

https://archive.ics.uci.edu/ml/datasets/Iris/

After that we read the iris.data without header and then we will add (using ds.columns)

header for our further operation.

```
>>> import numpy as np
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> ds=pd.read_csv("C:\\Users\santu modak\Downloads\iris.data",header=None)
>>> ds.columns=['Sepal.Length','Sepal.Width','Petal.Length','Petal.Width','Species']
>>> ds.head()
   Sepal.Length  Sepal.Width  Petal.Length  Petal.Width      Species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
>>>
```

Then we will divide the dataset as test data 25% and train data 75% by writing size=0.25

```
>>> from sklearn.model_selection import train_test_split
>>> train,test=train_test_split(ds,test_size=0.25)
>>> train
    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width          Species
33           5.5          4.2           1.4          0.2      Iris-setosa
50           7.0          3.2           4.7          1.4  Iris-versicolor
11           4.8          3.4           1.6          0.2      Iris-setosa
47           4.6          3.2           1.4          0.2      Iris-setosa
51           6.4          3.2           4.5          1.5  Iris-versicolor
..           ...          ...           ...          ...              ...
3            4.6          3.1           1.5          0.2      Iris-setosa
45           4.8          3.0           1.4          0.3      Iris-setosa
20           5.4          3.4           1.7          0.2      Iris-setosa
42           4.4          3.2           1.3          0.2      Iris-setosa
23           5.1          3.3           1.7          0.5      Iris-setosa

[112 rows x 5 columns]
>>>
```

Above picture data is train data and below picture data is test data

```
>>>
>>> test
     Sepal.Length  Sepal.Width  Petal.Length  Petal.Width         Species
44            5.1          3.8           1.9          0.4     Iris-setosa
135           7.7          3.0           6.1          2.3  Iris-virginica
76            6.8          2.8           4.8          1.4  Iris-versicolor
97            6.2          2.9           4.3          1.3  Iris-versicolor
117           7.7          3.8           6.7          2.2  Iris-virginica
137           6.4          3.1           5.5          1.8  Iris-virginica
136           6.3          3.4           5.6          2.4  Iris-virginica
102           7.1          3.0           5.9          2.1  Iris-virginica
22            4.6          3.6           1.0          0.2     Iris-setosa
128           6.4          2.8           5.6          2.1  Iris-virginica
48            5.3          3.7           1.5          0.2     Iris-setosa
119           6.0          2.2           5.0          1.5  Iris-virginica
141           6.9          3.1           5.1          2.3  Iris-virginica
10            5.4          3.7           1.5          0.2     Iris-setosa
143           6.8          3.2           5.9          2.3  Iris-virginica
36            5.5          3.5           1.3          0.2     Iris-setosa
82            5.8          2.7           3.9          1.2  Iris-versicolor
123           6.3          2.7           4.9          1.8  Iris-virginica
64            5.6          2.9           3.6          1.3  Iris-versicolor
131           7.9          3.8           6.4          2.0  Iris-virginica
37            4.9          3.1           1.5          0.1     Iris-setosa
40            5.0          3.5           1.3          0.3     Iris-setosa
63            6.1          2.9           4.7          1.4  Iris-versicolor
17            5.1          3.5           1.4          0.3     Iris-setosa
90            5.5          2.6           4.4          1.2  Iris-versicolor
7             5.0          3.4           1.5          0.2     Iris-setosa
52            6.9          3.1           4.9          1.5  Iris-versicolor
96            5.7          2.9           4.2          1.3  Iris-versicolor
71            6.1          2.8           4.0          1.3  Iris-versicolor
99            5.7          2.8           4.1          1.3  Iris-versicolor
38            4.4          3.0           1.3          0.2     Iris-setosa
140           6.7          3.1           5.6          2.4  Iris-virginica
26            5.0          3.4           1.6          0.4     Iris-setosa
138           6.0          3.0           4.8          1.8  Iris-virginica
94            5.6          2.7           4.2          1.3  Iris-versicolor
81            5.5          2.4           3.7          1.0  Iris-versicolor
109           7.2          3.6           6.1          2.5  Iris-virginica
35            5.0          3.2           1.2          0.2     Iris-setosa
>>>
```

After that We will divide some column as test_X and one column as test_Y for effective mapping.

```
>>>
>>> train_Y=train.Species
>>> train_Y
33         Iris-setosa
50     Iris-versicolor
11         Iris-setosa
47         Iris-setosa
51     Iris-versicolor
              ...
3          Iris-setosa
45         Iris-setosa
20         Iris-setosa
42         Iris-setosa
23         Iris-setosa
Name: Species, Length: 112, dtype: object
```

```
>>> train_X=train[['Sepal.Length','Sepal.Width','Petal.Length','Petal.Width']]
>>> train_X
    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
33           5.5          4.2           1.4          0.2
50           7.0          3.2           4.7          1.4
11           4.8          3.4           1.6          0.2
47           4.6          3.2           1.4          0.2
51           6.4          3.2           4.5          1.5
..           ...          ...           ...          ...
3            4.6          3.1           1.5          0.2
45           4.8          3.0           1.4          0.3
20           5.4          3.4           1.7          0.2
42           4.4          3.2           1.3          0.2
23           5.1          3.3           1.7          0.5

[112 rows x 4 columns]
>>>
```

Similarly we will divide test data as test_X and test_Y .we will give test data to the train model and matching with test_Y data .So that we can analysis the output.

```
>>>
>>> test_X=test[['Sepal.Length','Sepal.Width','Petal.Length','Petal.Width']]
>>> test_X
    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
44           5.1          3.8           1.9          0.4
135          7.7          3.0           6.1          2.3
76           6.8          2.8           4.8          1.4
97           6.2          2.9           4.3          1.3
117          7.7          3.8           6.7          2.2
137          6.4          3.1           5.5          1.8
136          6.3          3.4           5.6          2.4
102          7.1          3.0           5.9          2.1
22           4.6          3.6           1.0          0.2
128          6.4          2.8           5.6          2.1
48           5.3          3.7           1.5          0.2
119          6.0          2.2           5.0          1.5
141          6.9          3.1           5.1          2.3
10           5.4          3.7           1.5          0.2
143          6.8          3.2           5.9          2.3
36           5.5          3.5           1.3          0.2
32           5.8          2.7           3.9          1.2
123          6.3          2.7           4.9          1.8
54           5.6          2.9           3.6          1.3
131          7.9          3.8           6.4          2.0
37           4.9          3.1           1.5          0.1
40           5.0          3.5           1.3          0.3
53           6.1          2.9           4.7          1.4
17           5.1          3.5           1.4          0.3
90           5.5          2.6           4.4          1.2
7            5.0          3.4           1.5          0.2
52           6.9          3.1           4.9          1.5
96           5.7          2.9           4.2          1.3
71           6.1          2.8           4.0          1.3
99           5.7          2.8           4.1          1.3
38           4.4          3.0           1.3          0.2
140          6.7          3.1           5.6          2.4
26           5.0          3.4           1.6          0.4
138          6.0          3.0           4.8          1.8
94           5.6          2.7           4.2          1.3
31           5.5          2.4           3.7          1.0
109          7.2          3.6           6.1          2.5
35           5.0          3.2           1.2          0.2
>>>
```

```
>> test_Y=test.Species
>> test_Y
4         Iris-setosa
35     Iris-virginica
6     Iris-versicolor
7     Iris-versicolor
17     Iris-virginica
37     Iris-virginica
36     Iris-virginica
02     Iris-virginica
2         Iris-setosa
28     Iris-virginica
8         Iris-setosa
19     Iris-virginica
41     Iris-virginica
0         Iris-setosa
43     Iris-virginica
6         Iris-setosa
2     Iris-versicolor
23     Iris-virginica
4     Iris-versicolor
31     Iris-virginica
7         Iris-setosa
0         Iris-setosa
3     Iris-versicolor
7         Iris-setosa
0     Iris-versicolor
          Iris-setosa
2     Iris-versicolor
6     Iris-versicolor
1     Iris-versicolor
9     Iris-versicolor
8         Iris-setosa
40     Iris-virginica
6         Iris-setosa
38     Iris-virginica
```

In this example we use DecisionTreeClassifier without parameter tuning and use train_ X data and train_Y data to train the model. After that we will get the corresponding output by giving test_X dataand that will store in Y_pred.

```
>>> from sklearn.tree import DecisionTreeClassifier
>>> classifier=DecisionTreeClassifier()
>>> classifier.fit(train_X,train_Y)
DecisionTreeClassifier()
>>> Y_pred=classifier.predict(test_X)
>>>
```

OUTPUT WITHOUT PARAMETER TUNING:

Then we will check the performance of the model by analysis the Accuracy, Precision, Recall, F-score, confusion matrix output .

```
>>>
>>> from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(Y_pred,test_Y))
0.9736842105263158
>>> print("Precision, Recall, F-score")
Precision, Recall, F-score
>>> print(classification_report(Y_pred,test_Y))
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.97        38
      macro avg       0.98      0.97      0.97        38
   weighted avg       0.98      0.97      0.97        38

>>> print("confusion matrix")
confusion matrix
>>> print(confusion_matrix(Y_pred,test_Y))
[[12  0  0]
 [ 0 12  1]
 [ 0  0 13]]
>>>
```

OUTPUT WITH PARAMETER TUNING:

Here we will pass the parameter criterion =”entropy”(default “gini”)

Max_depth=3

```
>>> classifier=DecisionTreeClassifier(criterion="entropy",max_depth=3)
>>> classifier.fit(train_X,train_Y)
DecisionTreeClassifier(criterion='entropy', max_depth=3)
>>> Y_pred=classifier.predict(test_X)
```

```
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(Y_pred,test_Y)
... 
KeyboardInterrupt
>>> print(accuracy_score(Y_pred,test_Y))
0.9736842105263158
>>> print("Precision, Recall, F-score")
Precision, Recall, F-score
>>> print(classification_report(Y_pred,test_Y))
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.97        38
      macro avg       0.98      0.97      0.97        38
   weighted avg       0.98      0.97      0.97        38

>>> print("confusion matrix")
confusion matrix
>>> print(confusion_matrix(Y_pred,test_Y))
[[12  0  0]
 [ 0 12  1]
 [ 0  0 13]]
>>>
```

Here we will pass the parameter criterion =”entropy”(default “gini”)

Max_depth=10

```
>>> classifier=DecisionTreeClassifier(criterion="entropy",max_depth=10)
>>> classifier.fit(train_X,train_Y)
DecisionTreeClassifier(criterion='entropy', max_depth=10)
>>> pred=classifier.predict(test_X)
>>>
```

```
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(pred,test_Y)
...
KeyboardInterrupt
>>> print(accuracy_score(pred,test_Y))
0.9736842105263158
>>> print("Precision, Recall, F-score")
Precision, Recall, F-score
>>> print(classification_report(pred,test_Y))
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.97        38
      macro avg       0.98      0.97      0.97        38
   weighted avg       0.98      0.97      0.97        38

>>> print("confusion matrix")
confusion matrix
>>> print(confusion_matrix(pred,test_Y))
[[12  0  0]
 [ 0 12  1]
 [ 0  0 13]]
>>>
```

Here we will pass the parameter criterion =“gini”

Max_depth=10

```
>>>
>>> classifier=DecisionTreeClassifier(criterion="gini",max_depth=10)
>>> classifier.fit(train_X,train_Y)
DecisionTreeClassifier(max_depth=10)
>>> pred1=classifier.predict(test_X)
>>> _
```

```
>>>
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(pred1,test_Y))
0.9736842105263158
>>> print("Precision, Recall, F-score")
Precision, Recall, F-score
>>> print(classification_report(pred1,test_Y))
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.93      1.00      0.96        13

       accuracy                           0.97        38
      macro avg       0.98      0.97      0.97        38
   weighted avg       0.98      0.97      0.97        38

>>> print("confusion matrix")
confusion matrix
>>> print(confusion_matrix(pred1,test_Y))
[[12  0  0]
 [ 0 12  1]
 [ 0  0 13]]
>>>
```

Here we will pass the parameter criterion = “gini”

Max_depth=15

```
>>> classifier=DecisionTreeClassifier(criterion="gini",max_depth=15)
>>> classifier.fit(train_X,train_Y)
DecisionTreeClassifier(max_depth=15)
>>> pred2=classifier.predict(test_X)
>>>
```

```
>>>
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(pred2,test_Y))
0.9473684210526315
>>>
>>> print("Precision, Recall, F-score")
Precision, Recall, F-score
>>> print(classification_report(pred2,test_Y))
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       1.00      0.86      0.92        14
 Iris-virginica       0.86      1.00      0.92        12

       accuracy                           0.95        38
      macro avg       0.95      0.95      0.95        38
   weighted avg       0.95      0.95      0.95        38

>>>
>>>
>>> print("confusion matrix")
confusion matrix
>>> print(confusion_matrix(pred2,test_Y))
[[12  0  0]
 [ 0 12  2]
 [ 0  0 12]]
>>>
```

1.Naive Bayes:

There are three types of Naïve Bayes model under the scikit-learn library.

1. Here first we will use MultinomialNB classifier and calculate Accuracy, Precision, Recall, F-score, confusion matrix without parameter tuning.

```
>>>
>>> from sklearn.naive_bayes import MultinomialNB
>>> classifier=MultinomialNB().fit(train_X,train_Y)
>>> classifier.fit(train_X,train_Y)
MultinomialNB()
>>> pred=classifier.predict(test_X)
>>>
>>>
```

```
>>>
>>> from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(pred,test_Y)
...
KeyboardInterrupt
>>> print(accuracy_score(pred,test_Y))
0.8947368421052632
>>> print("Precision, Recall, F-score")
Precision, Recall, F-score
>>> print(classification_report(pred,test_Y))
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       1.00      0.75      0.86        16
 Iris-virginica       0.71      1.00      0.83        10

       accuracy                           0.89        38
      macro avg       0.90      0.92      0.90        38
   weighted avg       0.92      0.89      0.90        38

>>> print("confusion matrix")
confusion matrix
>>> print(confusion_matrix(pred,test_Y))
[[12  0  0]
 [ 0 12  4]
 [ 0  0 10]]
>>>
```

Output of MultinomialNB classifier with parameter tuning.

```
>>>
>>> classifier=MultinomialNB(alpha=2.5,fit_prior=True,class_prior=None).fit(train_X,train_Y)
>>> classifier.fit(train_X,train_Y)
MultinomialNB(alpha=2.5)
>>> pred=classifier.predict(test_X)
>>>
>>>
```

```
>>>
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(pred,test_Y))
0.8947368421052632
>>> print("Precision, Recall, F-score")
Precision, Recall, F-score
>>> print(classification_report(pred,test_Y))
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       1.00      0.75      0.86        16
 Iris-virginica       0.71      1.00      0.83        10

       accuracy                           0.89        38
      macro avg       0.90      0.92      0.90        38
   weighted avg       0.92      0.89      0.90        38

>>> print("confusion matrix")
confusion matrix
>>> print(confusion_matrix(pred,test_Y))
[[12  0  0]
 [ 0 12  4]
 [ 0  0 10]]
>>>
```

2. Now second type of classifier that we will use GaussianNB classifier and calculate Accuracy, Precision, Recall, F-score, confusion matrix without parameter tuning.

```
>>>
>>> from sklearn.naive_bayes import GaussianNB
>>> classifier= GaussianNB().fit(train_X,train_Y)
>>> classifier.fit(train_X,train_Y)
GaussianNB()
>>> y_pred=classfier.predict(test_X)
```

Output of GaussianNB classifier without parameter tuning

```
>>>
>>> print(accuracy_score(y_pred,test_Y))
0.9736842105263158
>>>
>>>
>>> print("precision,Recall,F-score")
precision,Recall,F-score
>>>
>>> print(classification_report(y_pred,test_Y))
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.94      0.97        16
 Iris-virginica       0.89      1.00      0.94         8

       accuracy                           0.97        38
      macro avg       0.96      0.98      0.97        38
   weighted avg       0.98      0.97      0.97        38

>>>
>>> print("Confusion matrix")
Confusion matrix
>>>
>>> print(confusion_matrix(y_pred,test_Y))
[[14  0  0]
 [ 0 15  1]
 [ 0  0  8]]
>>>
```

Output of GaussianNB classifier with parameter tuning.

```
>>>
>>> classifier= GaussianNB(priors=None,var_smoothing=1e-05).fit(train_X,train_Y)
>>> classifier.fit(train_X,train_Y)
GaussianNB(var_smoothing=1e-05)
>>> y_pred=classifier.predict(test_X)
>>>
```

```
>>>
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(y_pred,test_Y))
0.9736842105263158
>>>
>>> print("precision,Recall,F-score")
precision,Recall,F-score
>>>
>>> print(classification_report(y_pred,test_Y))
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.94      0.97        16
 Iris-virginica       0.89      1.00      0.94         8

       accuracy                           0.97        38
      macro avg       0.96      0.98      0.97        38
   weighted avg       0.98      0.97      0.97        38

>>> print("Confusion matrix")
Confusion matrix
>>>
>>> print(confusion_matrix(y_pred,test_Y))
[[14  0  0]
 [ 0 15  1]
 [ 0  0  8]]
>>>
```

3. Third type of classifier that we will use BernoulliNB classifier and calculate Accuracy, Precision, Recall, F-score, confusion matrix without parameter tuning.

```
>>>
>>> from sklearn.naive_bayes import BernoulliNB
>>> classifier=BernoulliNB().fit(train_X,train_Y)
>>> classifier.fit(train_X,train_Y)
BernoulliNB()
>>> y_pred=classifier.predict(test_X)
>>>
>>>
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(y_pred,test_Y))
0.23684210526315788
```

Output of BernoulliNB classifier without parameter tuning

```
                 precision    recall  f1-score   support

    Iris-setosa       0.00      0.00      0.00         0
Iris-versicolor       0.00      0.00      0.00         0
 Iris-virginica       1.00      0.24      0.38        38

       accuracy                           0.24        38
      macro avg       0.33      0.08      0.13        38
   weighted avg       1.00      0.24      0.38        38

>>>
>>> print("Confusion matrix")
Confusion matrix
>>> print(confusion_matrix(y_pred,test_Y))
[[ 0  0  0]
 [ 0  0  0]
 [14 15  9]]
>>>
```

Output of BernoulliNB classifier with parameter tuning

```
>>> classifier=BernoulliNB(alpha=1.0,binarize=0.0,fit_prior=True,class_prior=None).fit(train_X,train_Y)
>>>
>>> classifier.fit(train_X,train_Y)
BernoulliNB()
>>> y_pred=classifier.predict(test_X)
```

```
>>> print("Accuracy")
Accuracy
>>> print(accuracy_score(y_pred,test_Y))
0.23684210526315788
>>> print("precision,Recall,F-score")
precision,Recall,F-score
>>> print(classification_report(y_pred,test_Y))
C:\Users\santu modak\AppData\Local\Programs\Python\Python38-32\lib\site-packages\sklearn\metrics\_classi
: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\santu modak\AppData\Local\Programs\Python\Python38-32\lib\site-packages\sklearn\metrics\_classi
: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\santu modak\AppData\Local\Programs\Python\Python38-32\lib\site-packages\sklearn\metrics\_classi
: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true
ero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
                 precision    recall  f1-score   support

    Iris-setosa       0.00      0.00      0.00         0
Iris-versicolor       0.00      0.00      0.00         0
 Iris-virginica       1.00      0.24      0.38        38

       accuracy                           0.24        38
      macro avg       0.33      0.08      0.13        38
   weighted avg       1.00      0.24      0.38        38
```

```
>>
>> print("Confusion matrix")
onfusion matrix
>> print(confusion_matrix(y_pred,test_Y))
[ 0  0  0]
[ 0  0  0]
[14 15  9]]
>>
```

Diabetes Dataset

Q1.

Without parameter tuning:-

1. **Multinomial:--**
   Code:-

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Dataset Preparation

dataset = pd. read_csv (r"C:\Users\patra\Machine learning
Lab\datasets\diabetes.csv")
X = dataset.drop (['AGE', 'SEX'], axis=1)

y = dataset ['SEX']

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)

# Classification

from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB ().fit (X_train, y_train)
```

```
classifier.fit (X_train, y_train)

y_pred=classifier.predict (X_test)

# Evaluation of Classifier Performance

from sklearn.metrics import classification_report, confusion_matrix

print("Confusion Matrix:")

print (confusion_matrix (y_test, y_pred))

print("-----------------------------------------")

print ("-----------------------------------------")

print("Performance Evaluation:")

print (classification_report (y_test, y_pred))
```

Output:-

```
PS C:\Users\patra\Machine learning Lab> &
C:/Users/patra/AppData/Local/Programs/Python/Python39/python.exe "
C:/Users/patra/Machine learning Lab/code/lab assignment1/Q1_Naive_Bayes_Diabetes.py"
Confusion Matrix:
[[27 15]
 [14 33]]
-----------------------------------------
-----------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           1       0.66      0.64      0.65        42
           2       0.69      0.70      0.69        47

    accuracy                           0.67        89
   macro avg       0.67      0.67      0.67        89
weighted avg       0.67      0.67      0.67        89
```

2.Gaussian:--

Code:-

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB ().fit (X_train, y_train)
classifier.fit (X_train, y_train)

y_pred=classifier.predict (X_test)
```

Output:-

PS C:\Users\patra\Machine learning Lab> &
C:/Users/patra/AppData/Local/Programs/Python/Python39/python.exe "
c:/Users/patra/Machine learning Lab/code/lab assignment1/Q1_Naive_Bayes_Diabetes.py"
Confusion Matrix:
[[28 15]
 [14 32]]

Performance Evaluation:

	precision	recall	f1-score	support
1	0.67	0.65	0.66	43
2	0.68	0.70	0.69	46
accuracy			0.67	89
macro avg	0.67	0.67	0.67	89
weighted avg	0.67	0.67	0.67	89

3. Bernoulli:---

Code:-

```
from sklearn.naive_bayes import BernoulliNB
classifier = BernoulliNB ().fit (X_train, y_train)
classifier.fit (X_train, y_train)

y_pred=classifier.predict (X_test)
```

Output:-
Confusion Matrix:
[[48 0]
 [41 0]]

Performance Evaluation:

	precision	recall	f1-score	support
1	0.54	1.00	0.70	48
2	0.00	0.00	0.00	41
accuracy			0.54	89
macro avg	0.27	0.50	0.35	89
weighted avg	0.29	0.54	0.38	89

2. Parameter tuning:---

1.Multonomial:-

Using `alpha=2.5,fit_prior=True,class_prior=None :---`

Code:-

```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB (alpha=2.5,fit_prior=True,class_prior=None).fit
(X_train, y_train)
classifier.fit (X_train, y_train)

y_pred=classifier.predict (X_test)
```

Output:-

PS C:/Users/patraMachine learning Lab> &
C:/Users/patra/AppData/Local/Programs/Python/Python39/python.exe"
c:/Users/patra/Machine learning Lab/code/lab assignment1/Q1_Naive_Bayes_Diabetes.py"
Confusion Matrix:
[[23 18]
 [16 32]]

Performance Evaluation:

	precision	recall	f1-score	support

1	0.59	0.56	0.57	41
2	0.64	0.67	0.65	48
accuracy			0.62	89
macro avg	0.61	0.61	0.61	89
weighted avg	0.62	0.62	0.62	89

2.Gaussian:-

Using priors=None:---

Code:-

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB (priors=None).fit (X_train, y_train)
classifier.fit (X_train, y_train)

y_pred=classifier.predict (X_test)
```

Output:-

PS C:\Users\patra\Machine learning Lab> &
C:/Users/patra/AppData/Local/Programs/Python/Python39/python.exe “
c:/Users/patra/Machine learning Lab/code/lab assignment1/Q1_Naive_Bayes_Diabetes.py"

Confusion Matrix:

[[31 15]
 [14 29]]

Performance Evaluation:

	precision	recall	f1-score	support
1	0.69	0.67	0.68	46
2	0.66	0.67	0.67	43
accuracy			0.67	89
macro avg	0.67	0.67	0.67	89
weighted avg	0.67	0.67	0.67	89

3.Bernoulli:-

Using `alpha=1.0,binarize=0.0,fit_prior=True,class_prior=None:--`

Code:-

```
from sklearn.naive_bayes import BernoulliNB
classifier = BernoulliNB
(alpha=1.0,binarize=0.0,fit_prior=True,class_prior=None).fit (X_train,
y_train)
classifier.fit (X_train, y_train)

y_pred=classifier.predict (X_test)
```

Output:-

PS C:\Users\patra\Machine learning Lab> &
C:/Users/patra/AppData/Local/Programs/Python/Python39/python.exe "
c:/Users/patra/Machine learning Lab/code/lab assignment1/Q1_Naive_Bayes_Diabetes.py"
Confusion Matrix:
[[47 0]
[42 0]]

Performance Evaluation:

	precision	recall	f1-score	support
1	0.53	1.00	0.69	47
2	0.00	0.00	0.00	42
accuracy			0.53	89
macro avg	0.26	0.50	0.35	89
weighted avg	0.28	0.53	0.37	89

Q2.

Without parameter tuning:-------

Code:---

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

# Dataset Preparation

dataset = pd.read_csv(r"C:\Users\patra\Machine learning
Lab\datasets\diabetes.csv")
X = dataset.drop (['AGE', 'SEX'], axis=1)
y = dataset ['SEX']

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)

# Classification

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier()
```

```
classifier.fit (X_train, y_train)

y_pred=classifier.predict (X_test)

# Evaluation of classifier Performance

from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")

print (confusion_matrix(y_test, y_pred))

print("----------------------------------------------------------")
print("----------------------------------------------------------")

print("themance Evaluation:")
print (classification_report (y_test, y_pred))
```

Output:-

PS C:\Users\patra\Machine learning Lab> &
C:/Users/patra/AppData/Local/Programs/Python/Python39/python.exe "
c:/Users/patra/Machine learning Lab/code/lab assignment1/Q2_Decision_tree_diabetes.py"
Confusion Matrix:
[[24 26]
 [9 30]]

---
themance Evaluation:

	precision	recall	f1-score	support
1	0.73	0.48	0.58	50
2	0.54	0.77	0.63	39
accuracy			0.61	89
macro avg	0.63	0.62	0.60	89
weighted avg	0.64	0.61	0.60	89

Parameter tuning:-

1.Making criterion=”gini” and max_depth=10

```
from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion="gini",max_depth=10)
classifier.fit (X_train, y_train)

y_pred=classifier.predict (X_test)
```

Output:-

PS C:\Users\patra\Machine learning Lab> &
C:/Users/patra/AppData/Local/Programs/Python/Python39/python.exe”
c:/Users/patra/Machine learning Lab/code/lab assignment1/Q2_Decision_tree_diabetes.py"
Confusion Matrix:
[[26 21]
 [16 26]]

---
themance Evaluation:

	precision	recall	f1-score	support
1	0.62	0.55	0.58	47
2	0.55	0.62	0.58	42
accuracy			0.58	89
macro avg	0.59	0.59	0.58	89
weighted avg	0.59	0.58	0.58	89

2.Making criterion=”entropy” and max_depth=10

Code:-

```
from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion="entropy",max_depth=10)
classifier.fit (X_train, y_train)
```

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

Output:-

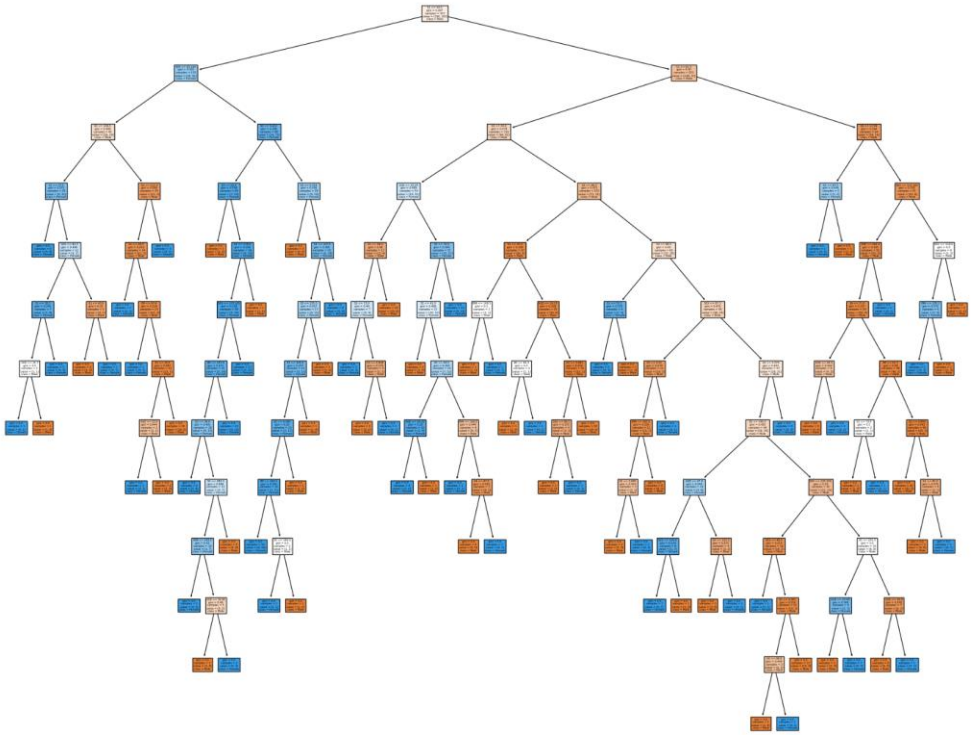
PS C:\Users\patra\Machine learning Lab> &
C:/Users/patra/AppData/Local/Programs/Python/Python39/python.exe ''
c:/Users/patra/Machine learning Lab/code/lab assignment1/Q2_Decision_tree_diabetes.py''
Confusion Matrix:
[[35 14]
 [20 20]]

---
themance Evaluation:

	precision	recall	f1-score	support
1	0.64	0.71	0.67	49
2	0.59	0.50	0.54	40
accuracy			0.62	89
macro avg	0.61	0.61	0.61	89
weighted avg	0.61	0.62	0.61	89

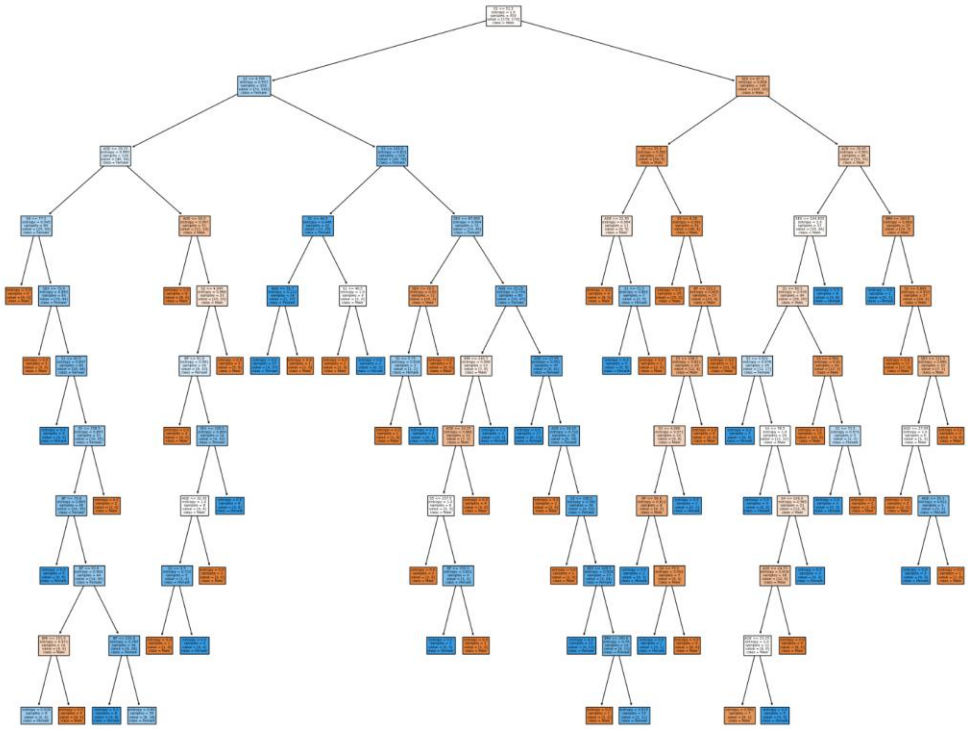
Images:-

1.Without parameter tuning:-



2.Using Parameter tuning:-

1.Making criterion=”gini” and max_depth=10.



2.Making criterion=”entropy” and max_depth=10.