README.md

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display, HTML
from sklearn.naive_bayes import BernoulliNB, MultinomialNB, GaussianNB
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_m
```

```
from sklearn.datasets import load_iris, load_breast_cancer
iris_dataset=load_iris()
breast_cancer_dataset=load_breast_cancer()
```

IRIS DATASET

```
df_iris=pd.DataFrame(iris_dataset.data,columns=iris_dataset.feature_names)
display(df_iris)
df_iris_target=iris_dataset.target
display(df_iris_target)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
array([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, 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
X_train,X_test,Y_train,Y_test=train_test_split(df_iris,df_iris_target,test_siz
```

```
dt_classifier=DecisionTreeClassifier()
dt_classifier.fit(X_train,Y_train)
Y_pred=dt_classifier.predict(X_test)
```

```
dt_cm=confusion_matrix(Y_test,Y_pred)
print("Confusion Matrix:")
print(dt_cm)
print("=========================================")
print("=========================================")
print("Performance Evaluation:")
print(classification_report(Y_test,Y_pred))
```

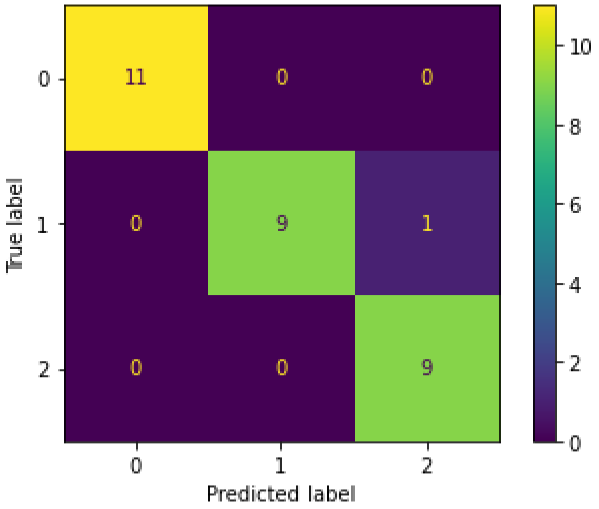
```
Confusion Matrix:
[[11  0  0]
 [ 0  9  1]
 [ 0  0  9]]
=========================================
=========================================
Performance Evaluation:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.90      0.95        10
           2       0.90      1.00      0.95         9

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

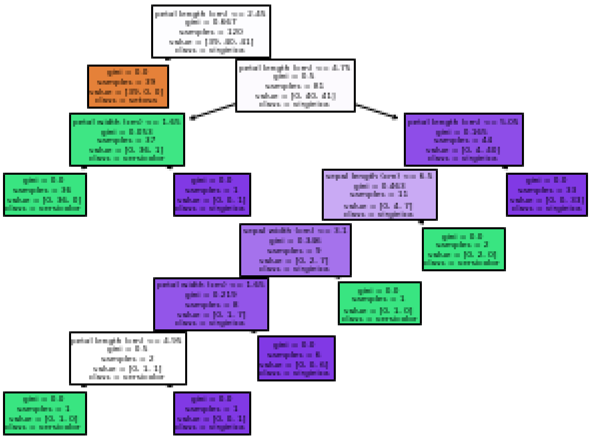
```
(ConfusionMatrixDisplay(confusion_matrix=dt_cm).plot())
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fbcf3cc45
```



```
from sklearn.tree import plot_tree
plot_tree(dt_classifier,feature_names=iris_dataset.feature_names,class_names=i
```

```
[Text(0.375, 0.9375, 'petal length (cm) <= 2.45\ngini = 0.667\nsamples = 120\n
 Text(0.25, 0.8125, 'gini = 0.0\nsamples = 39\nvalue = [39, 0, 0]\nclass = set
 Text(0.5, 0.8125, 'petal length (cm) <= 4.75\ngini = 0.5\nsamples = 81\nvalue
 Text(0.25, 0.6875, 'petal width (cm) <= 1.65\ngini = 0.053\nsamples = 37\nval
 Text(0.125, 0.5625, 'gini = 0.0\nsamples = 36\nvalue = [0, 36, 0]\nclass = ve
 Text(0.375, 0.5625, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]\nclass = virg
 Text(0.75, 0.6875, 'petal length (cm) <= 5.05\ngini = 0.165\nsamples = 44\nva
 Text(0.625, 0.5625, 'sepal length (cm) <= 6.5\ngini = 0.463\nsamples = 11\nva
 Text(0.5, 0.4375, 'sepal width (cm) <= 3.1\ngini = 0.346\nsamples = 9\nvalue
 Text(0.375, 0.3125, 'petal width (cm) <= 1.65\ngini = 0.219\nsamples = 8\nval
 Text(0.25, 0.1875, 'petal length (cm) <= 4.95\ngini = 0.5\nsamples = 2\nvalue
 Text(0.125, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = vers
 Text(0.375, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]\nclass = virg
 Text(0.5, 0.1875, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 6]\nclass = virgin
 Text(0.625, 0.3125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0]\nclass = vers
 Text(0.75, 0.4375, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0]\nclass = versi
 Text(0.875, 0.5625, 'gini = 0.0\nsamples = 33\nvalue = [0, 0, 33]\nclass = vi
```



```
mnb_classifier=MultinomialNB().fit(X_train,Y_train)
mnb_classifier.fit(X_train,Y_train)
Y_pred=mnb_classifier.predict(X_test)
```

```
mnb_cm=confusion_matrix(Y_test,Y_pred)
print("Confusion Matrix:")
print(mnb_cm)
print("=========================================")
print("=========================================")
print("Performance Evaluation:")
print(classification_report(Y_test,Y_pred))
```

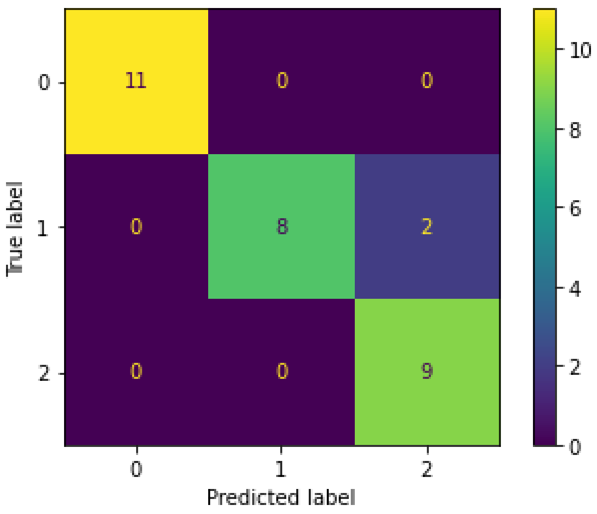
```
Confusion Matrix:
[[11  0  0]
 [ 0  8  2]
 [ 0  0  9]]
=========================================
=========================================
Performance Evaluation:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.80      0.89        10
           2       0.82      1.00      0.90         9

    accuracy                           0.93        30
   macro avg       0.94      0.93      0.93        30
weighted avg       0.95      0.93      0.93        30
```

```
(ConfusionMatrixDisplay(confusion_matrix=mnb_cm).plot())
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fbd0162ea
```



```
gnb_classifier=GaussianNB().fit(X_train,Y_train)
gnb_classifier.fit(X_train,Y_train)
Y_pred=gnb_classifier.predict(X_test)
```

```
gnb_cm=confusion_matrix(Y_test,Y_pred)
print("Confusion Matrix:")
print(gnb_cm)
print("=========================================")
print("=========================================")
print("Performance Evaluation:")
print(classification_report(Y_test,Y_pred))
```

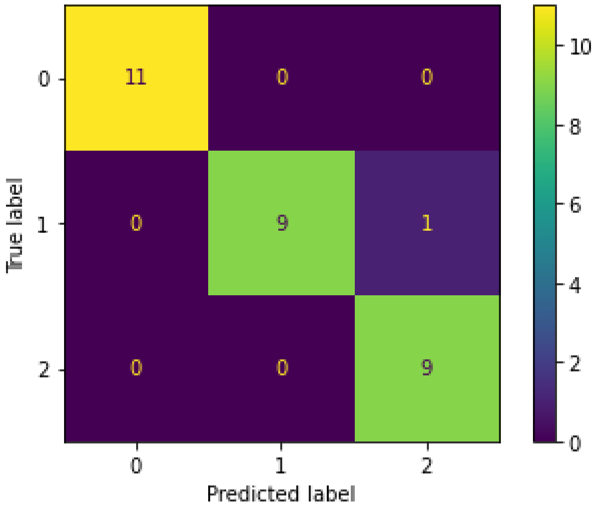
```
Confusion Matrix:
[[11  0  0]
 [ 0  9  1]
 [ 0  0  9]]
=========================================
=========================================
Performance Evaluation:
              precision    recall  f1-score   support
```

```
           0       1.00      1.00      1.00        11
           1       1.00      0.90      0.95        10
           2       0.90      1.00      0.95         9

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

```
(ConfusionMatrixDisplay(confusion_matrix=gnb_cm).plot())
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fbcf3e097
```



```
bnb_classifier=BernoulliNB(alpha=1.1, binarize=1.7).fit(X_train,Y_train)
bnb_classifier.fit(X_train,Y_train)
Y_pred=bnb_classifier.predict(X_test)
```

```
bnb_cm=confusion_matrix(Y_test,Y_pred)
print("Confusion Matrix:")
print(bnb_cm)
print("=========================================")
print("=========================================")
print("Performance Evaluation:")
print(classification_report(Y_test,Y_pred))
```

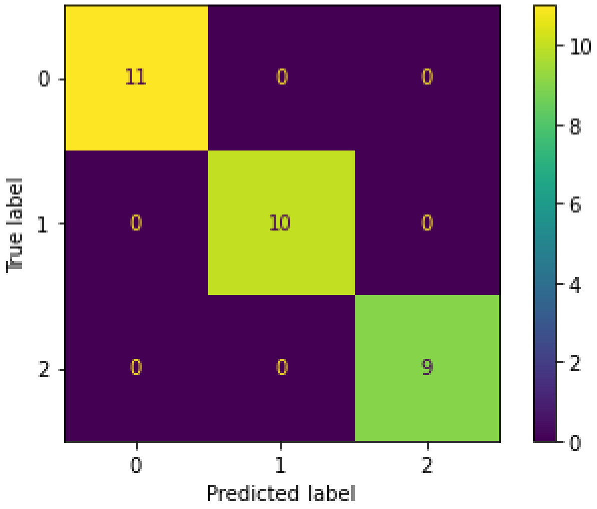
```
Confusion Matrix:
[[11  0  0]
 [ 0 10  0]
 [ 0  0  9]]
=========================================
=========================================
Performance Evaluation:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        10
           2       1.00      1.00      1.00         9

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

```
(ConfusionMatrixDisplay(confusion_matrix=bnb_cm).plot())
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fbcf3e7fc
```



BREAST CANCER DATASET

```
df_cancer=pd.DataFrame(breast_cancer_dataset.data,columns=breast_cancer_datase
display(df_cancer)
df_cancer_target=breast_cancer_dataset.target
display(df_cancer_target)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	11.42	20.38	77.58	386.1	0.14250	0.28390
4	20.29	14.34	135.10	1297.0	0.10030	0.13280
...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590
565	20.13	28.25	131.20	1261.0	0.09780	0.10340
566	16.60	28.08	108.30	858.1	0.08455	0.10230
567	20.60	29.33	140.10	1265.0	0.11780	0.27700
568	7.76	24.54	47.92	181.0	0.05263	0.04362

569 rows × 30 columns

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

```
X_train,X_test,Y_train,Y_test=train_test_split(df_cancer,df_cancer_target,test
```

```
dt_classifier=DecisionTreeClassifier()
dt_classifier.fit(X_train,Y_train)
Y_pred=dt_classifier.predict(X_test)
```

```
dt_cm=confusion_matrix(Y_test,Y_pred)
print("Confusion Matrix:")
print(dt_cm)
print("=========================================")
print("=========================================")
print("Performance Evaluation:")
print(classification_report(Y_test,Y_pred))
```

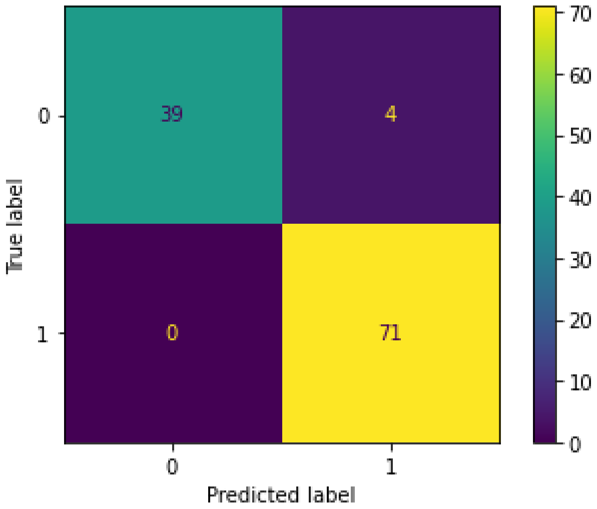
```
Confusion Matrix:
[[39  4]
 [ 0 71]]
=========================================
=========================================
Performance Evaluation:
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        43
           1       0.95      1.00      0.97        71

    accuracy                           0.96       114
   macro avg       0.97      0.95      0.96       114
weighted avg       0.97      0.96      0.96       114
```

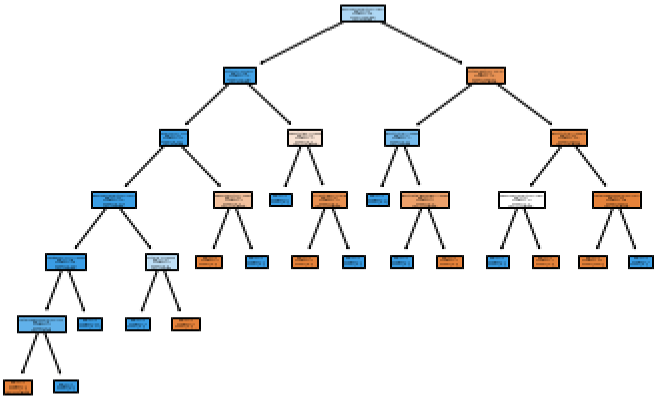
```
(ConfusionMatrixDisplay(confusion_matrix=dt_cm).plot())
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fbd017e19
```



```
plot_tree(dt_classifier,feature_names=breast_cancer_dataset.feature_names,clas
```

```
[Text(0.5491071428571429, 0.9285714285714286, 'mean concave points <= 0.052\ng
 Text(0.36607142857142855, 0.7857142857142857, 'worst radius <= 16.83\ngini =
 Text(0.26785714285714285, 0.6428571428571429, 'area error <= 48.7\ngini = 0.0
 Text(0.17857142857142858, 0.5, 'mean concave points <= 0.049\ngini = 0.023\ns
 Text(0.10714285714285714, 0.35714285714285715, 'smoothness error <= 0.003\ngi
 Text(0.07142857142857142, 0.21428571428571427, 'fractal dimension error <= 0.
 Text(0.03571428571428571, 0.07142857142857142, 'gini = 0.0\nsamples = 1\nvalu
 Text(0.10714285714285714, 0.07142857142857142, 'gini = 0.0\nsamples = 5\nvalu
 Text(0.14285714285714285, 0.21428571428571427, 'gini = 0.0\nsamples = 249\nva
 Text(0.25, 0.35714285714285715, 'worst area <= 796.25\ngini = 0.48\nsamples =
 Text(0.21428571428571427, 0.21428571428571427, 'gini = 0.0\nsamples = 3\nvalu
 Text(0.2857142857142857, 0.21428571428571427, 'gini = 0.0\nsamples = 2\nvalue
 Text(0.35714285714285715, 0.5, 'mean concavity <= 0.029\ngini = 0.444\nsample
 Text(0.32142857142857145, 0.35714285714285715, 'gini = 0.0\nsamples = 2\nvalu
 Text(0.39285714285714285, 0.35714285714285715, 'gini = 0.0\nsamples = 1\nvalu
 Text(0.4642857142857143, 0.6428571428571429, 'mean texture <= 18.68\ngini = 0
 Text(0.42857142857142855, 0.5, 'gini = 0.0\nsamples = 6\nvalue = [0, 6]\nclas
 Text(0.5, 0.5, 'concavity error <= 0.03\ngini = 0.18\nsamples = 10\nvalue = [
 Text(0.4642857142857143, 0.35714285714285715, 'gini = 0.0\nsamples = 9\nvalue
 Text(0.5357142857142857, 0.35714285714285715, 'gini = 0.0\nsamples = 1\nvalue
 Text(0.7321428571428571, 0.7857142857142857, 'worst perimeter <= 101.95\ngini
 Text(0.6071428571428571, 0.6428571428571429, 'worst texture <= 25.89\ngini =
 Text(0.5714285714285714, 0.5, 'gini = 0.0\nsamples = 12\nvalue = [0, 12]\ncla
 Text(0.6428571428571429, 0.5, 'worst fractal dimension <= 0.086\ngini = 0.32\
 Text(0.6071428571428571, 0.35714285714285715, 'gini = 0.0\nsamples = 1\nvalue
 Text(0.6785714285714286, 0.35714285714285715, 'gini = 0.0\nsamples = 4\nvalue
 Text(0.8571428571428571, 0.6428571428571429, 'worst texture <= 20.875\ngini =
 Text(0.7857142857142857, 0.5, 'mean concave points <= 0.091\ngini = 0.5\nsamp
 Text(0.75, 0.35714285714285715, 'gini = 0.0\nsamples = 7\nvalue = [0, 7]\ncla
 Text(0.8214285714285714, 0.35714285714285715, 'gini = 0.0\nsamples = 7\nvalue
 Text(0.9285714285714286, 0.5, 'fractal dimension error <= 0.013\ngini = 0.014
 Text(0.8928571428571429, 0.35714285714285715, 'gini = 0.0\nsamples = 144\nval
 Text(0.9642857142857143, 0.35714285714285715, 'gini = 0.0\nsamples = 1\nvalue
```



```
mnb_classifier=MultinomialNB(alpha=1.1).fit(X_train,Y_train)
mnb_classifier.fit(X_train,Y_train)
Y_pred=mnb_classifier.predict(X_test)
```

```
mnb_cm=confusion_matrix(Y_test,Y_pred)
print("Confusion Matrix:")
print(mnb_cm)
print("=========================================")
print("=========================================")
print("Performance Evaluation:")
print(classification_report(Y_test,Y_pred))
```

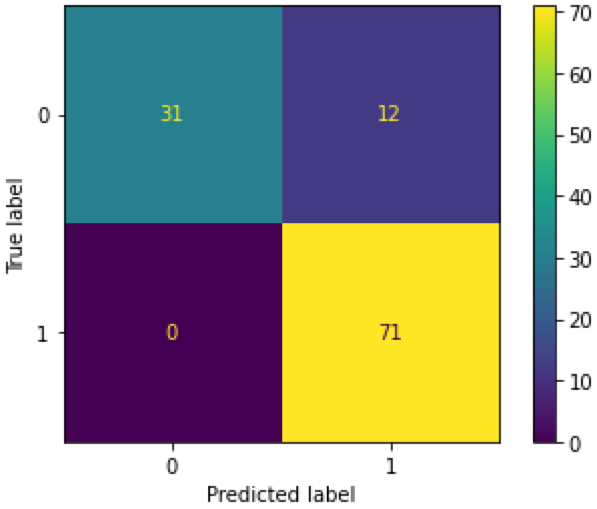
```
Confusion Matrix:
[[31 12]
 [ 0 71]]
=========================================
=========================================
Performance Evaluation:
              precision    recall  f1-score   support

           0       1.00      0.72      0.84        43
           1       0.86      1.00      0.92        71

    accuracy                           0.89       114
   macro avg       0.93      0.86      0.88       114
weighted avg       0.91      0.89      0.89       114
```

```
(ConfusionMatrixDisplay(confusion_matrix=mnb_cm).plot())
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fbcf3f9a5
```



```
gnb_classifier=GaussianNB().fit(X_train,Y_train)
gnb_classifier.fit(X_train,Y_train)
Y_pred=gnb_classifier.predict(X_test)
```

```
gnb_cm=confusion_matrix(Y_test,Y_pred)
print("Confusion Matrix:")
print(gnb_cm)
print("=========================================")
print("=========================================")
print("Performance Evaluation:")
print(classification_report(Y_test,Y_pred))
```

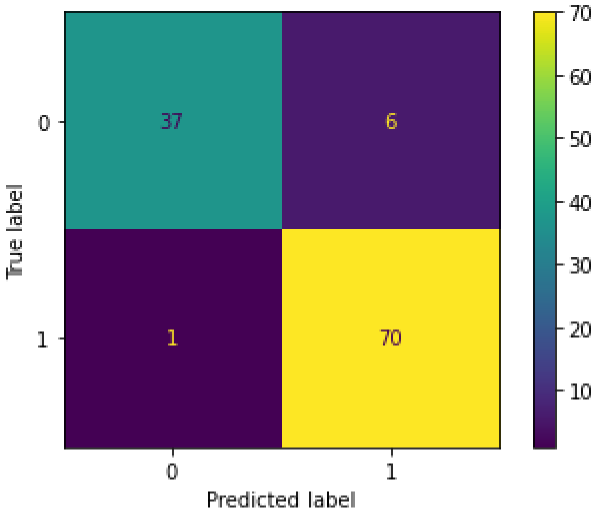
```
Confusion Matrix:
[[37  6]
 [ 1 70]]
=========================================
=========================================
Performance Evaluation:
              precision    recall  f1-score   support

           0       0.97      0.86      0.91        43
           1       0.92      0.99      0.95        71

    accuracy                           0.94       114
   macro avg       0.95      0.92      0.93       114
weighted avg       0.94      0.94      0.94       114
```

```
(ConfusionMatrixDisplay(confusion_matrix=gnb_cm).plot())
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fbd014e2d
```



```
bnb_cm=BernoulliNB(alpha=1.1, binarize=2.9).fit(X_train,Y_train)
bnb_cm.fit(X_train,Y_train)
Y_pred=bnb_cm.predict(X_test)
```

```
bnb_cm=confusion_matrix(Y_test,Y_pred)
print("Confusion Matrix:")
print(bnb_cm)
print("=========================================")
print("=========================================")
print("Performance Evaluation:")
print(classification_report(Y_test,Y_pred))
```

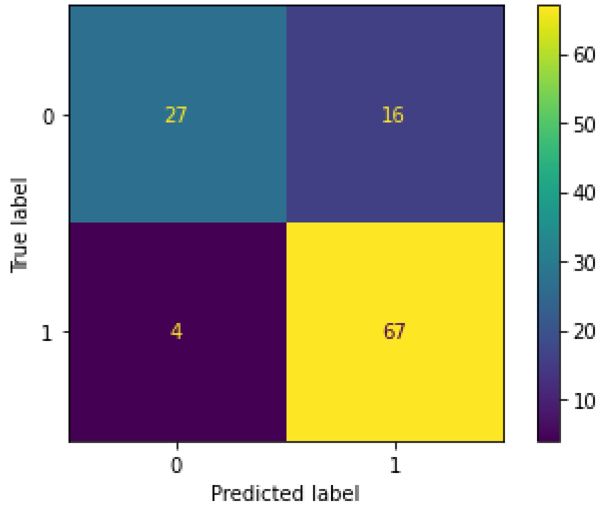
```
Confusion Matrix:
[[27 16]
 [ 4 67]]
=========================================
=========================================
Performance Evaluation:
              precision    recall  f1-score   support

           0       0.87      0.63      0.73        43
           1       0.81      0.94      0.87        71

    accuracy                           0.82       114
   macro avg       0.84      0.79      0.80       114
weighted avg       0.83      0.82      0.82       114
```

```
(ConfusionMatrixDisplay(confusion_matrix=bnb_cm).plot())
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fbce02f95
```



DIABETES DATASET

```
# Load the Iris Dataset
diabetes_data = pd.read_csv('./diabetes.csv')
display(diabetes_data)

# Separate features (X) and labels (y)
X = diabetes_data.drop('Outcome', axis=1)
y = diabetes_data['Outcome']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, rando
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BM
0	6	148	72	35	0	33.
1	1	85	66	29	0	26.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BM
2	8	183	64	0	0	23.
3	1	89	66	23	94	28.
4	0	137	40	35	168	43.
...	...	...	...	...	...	...
763	10	101	76	48	180	32.
764	2	122	70	27	0	36.
765	5	121	72	23	112	26.
766	1	126	60	0	0	30.
767	1	93	70	31	0	30.

768 rows × 9 columns

```
# Gaussian Naive Bayes Classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred_gnb = gnb.predict(X_test)

# Multinomial Naive Bayes Classifier
mnb = MultinomialNB()
mnb.fit(X_train, y_train)
y_pred_mnb = mnb.predict(X_test)

# Bernoulli Naive Bayes Classifier
bnb = BernoulliNB(alpha=1.1,binarize=1.1,fit_prior=True,class_prior=None)
bnb.fit(X_train, y_train)
y_pred_bnb = bnb.predict(X_test)

# Evaluate Gaussian Naive Bayes
accuracy_gnb = accuracy_score(y_test, y_pred_gnb)
print("Gaussian Naive Bayes")
print("=========================================")
print(f'Accuracy: {accuracy_gnb:.2f}')
print(classification_report(y_test, y_pred_gnb))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_gnb))

# Evaluate Multinomial Naive Bayes
accuracy_mnb = accuracy_score(y_test, y_pred_mnb)
print("\nMultinomial Naive Bayes")
print("=========================================")
print(f'Accuracy: {accuracy_mnb:.2f}')
print(classification_report(y_test, y_pred_mnb))
```

```python
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_mnb))

# Evaluate Bernoulli Naive Bayes
accuracy_bnb = accuracy_score(y_test, y_pred_bnb)
print("\nBernoulli Naive Bayes")
print("=========================================")
print(f'Accuracy: {accuracy_bnb:.2f}')
print(classification_report(y_test, y_pred_bnb))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_bnb))
```

```
Gaussian Naive Bayes
=========================================
Accuracy: 0.76
              precision    recall  f1-score   support

           0       0.83      0.81      0.82       206
           1       0.63      0.67      0.65       102

    accuracy                           0.76       308
   macro avg       0.73      0.74      0.73       308
weighted avg       0.76      0.76      0.76       308

Confusion Matrix:
[[166  40]
 [ 34  68]]

Multinomial Naive Bayes
=========================================
Accuracy: 0.61
              precision    recall  f1-score   support

           0       0.72      0.70      0.71       206
           1       0.42      0.44      0.43       102

    accuracy                           0.61       308
   macro avg       0.57      0.57      0.57       308
weighted avg       0.62      0.61      0.62       308

Confusion Matrix:
[[144  62]
 [ 57  45]]

Bernoulli Naive Bayes
=========================================
Accuracy: 0.69
              precision    recall  f1-score   support

           0       0.68      0.99      0.81       206
           1       0.73      0.08      0.14       102
    accuracy                           0.69       308
   macro avg       0.71      0.53      0.47       308
weighted avg       0.70      0.69      0.59       308

Confusion Matrix:
[[203   3]
 [ 94   8]]
```

```
from sklearn.tree import DecisionTreeClassifier, export_text, export_graphviz
import pydotplus
from IPython.display import Image

# Decision Tree Classifier
classifier = DecisionTreeClassifier(criterion='entropy')  # You can use 'gini'
classifier.fit(X_train, y_train)
y_pred_dtc = classifier.predict(X_test)

# Evaluate Decision Tree Classifier
accuracy_dtc = accuracy_score(y_test, y_pred_dtc)
print("Decision Tree Classifier:")
print(f'Accuracy: {accuracy_dtc:.2f}')
print(classification_report(y_test, y_pred_dtc))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_dtc))

dot_data = export_graphviz(classifier, out_file=None, feature_names=X.columns,
                           filled=True, rounded=True, special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)

# Save the decision tree image
decision_tree_image_path = "decision_tree.png"
graph.write_png(decision_tree_image_path)

# Display the decision tree image
Image(decision_tree_image_path)
```

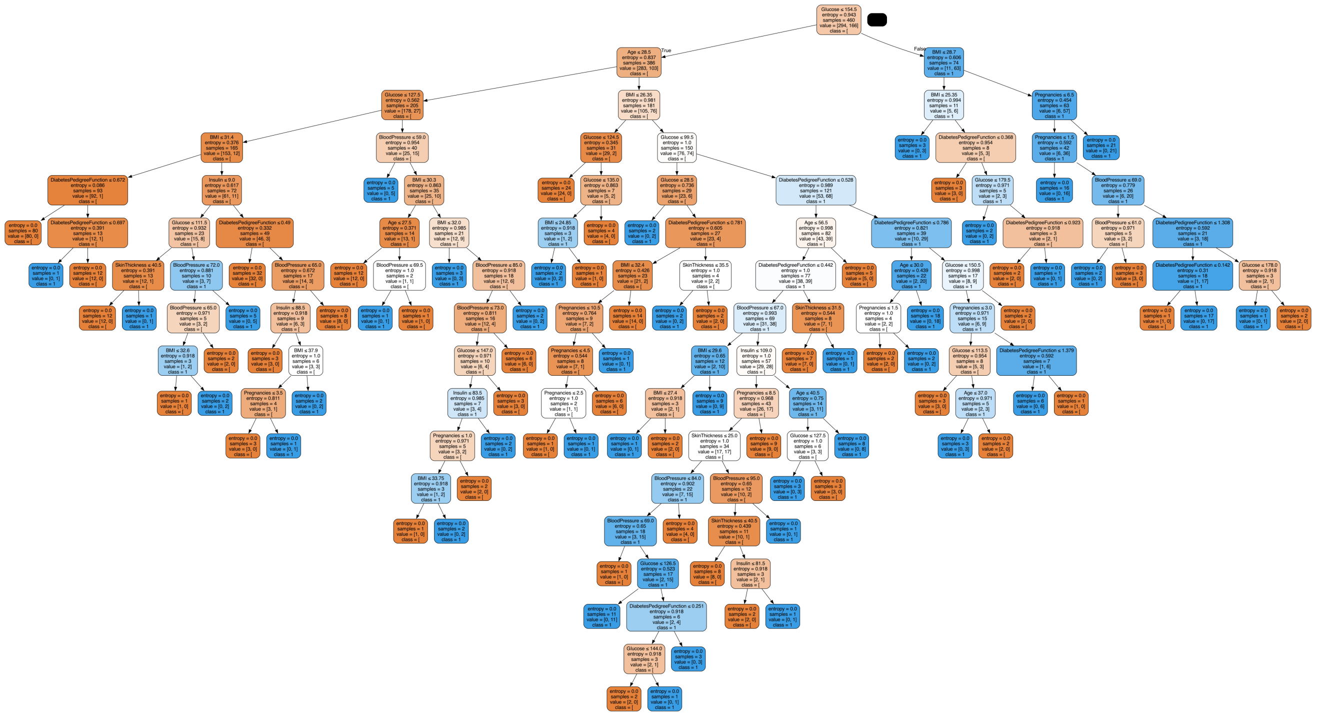
```
Decision Tree Classifier:
Accuracy: 0.73
              precision    recall  f1-score   support

           0       0.82      0.76      0.79       206
           1       0.58      0.67      0.62       102

    accuracy                           0.73       308
   macro avg       0.70      0.71      0.71       308
weighted avg       0.74      0.73      0.73       308

Confusion Matrix:
```

```
[[157  49]
 [ 34  68]]
```


Glucose ≤ 154.5 entropy = 0.943 samples = 460 value = [294, 166] class = [
[redacted]
True
False
Age ≤ 28.5 entropy = 0.837 samples = 386 value = [283, 103] class = [
BMI ≤ 28.7 entropy = 0.606 samples = 74 value = [11, 63] class = 1
Glucose ≤ 127.5 entropy = 0.562 samples = 205 value = [178, 27] class = [
BMI ≤ 26.35 entropy = 0.981 samples = 181 value = [105, 76] class = [
BMI ≤ 25.35 entropy = 0.994 samples = 11 value = [5, 6] class = 1
Pregnancies ≤ 6.5 entropy = 0.454 samples = 63 value = [6, 57] class = 1
BMI ≤ 31.4 entropy = 0.376 samples = 165 value = [153, 12] class = [
BloodPressure ≤ 59.0 entropy = 0.954 samples = 40 value = [25, 15] class = [
Glucose ≤ 124.5 entropy = 0.345 samples = 31 value = [29, 2] class = [
Glucose ≤ 99.5 entropy = 1.0 samples = 150 value = [76, 74] class = [
entropy = 0.0 samples = 3 value = [0, 3] class = 1
DiabetesPedigreeFunction ≤ 0.368 entropy = 0.954 samples = 8 value = [5, 3] class = [
Pregnancies ≤ 1.5 entropy = 0.592 samples = 42 value = [6, 36] class = 1
entropy = 0.0 samples = 21 value = [0, 21] class = 1
DiabetesPedigreeFunction ≤ 0.672 entropy = 0.086 samples = 93 value = [92, 1] class = [
Insulin ≤ 9.0 entropy = 0.617 samples = 72 value = [61, 11] class = [
entropy = 0.0 samples = 5 value = [0, 5] class = 1
BMI ≤ 30.3 entropy = 0.863 samples = 35 value = [25, 10] class = [
entropy = 0.0 samples = 24 value = [24, 0] class = [
Glucose ≤ 135.0 entropy = 0.863 samples = 7 value = [5, 2] class = [
Glucose ≤ 28.5 entropy = 0.736 samples = 29 value = [23, 6] class = [
DiabetesPedigreeFunction ≤ 0.528 entropy = 0.989 samples = 121 value = [53, 68] class = 1
entropy = 0.0 samples = 3 value = [3, 0] class = [
Glucose ≤ 179.5 entropy = 0.971 samples = 5 value = [2, 3] class = 1
entropy = 0.0 samples = 16 value = [0, 16] class = 1
BloodPressure ≤ 69.0 entropy = 0.779 samples = 26 value = [6, 20] class = 1
entropy = 0.0 samples = 80 value = [80, 0] class = [
DiabetesPedigreeFunction ≤ 0.697 entropy = 0.391 samples = 13 value = [12, 1] class = [
Glucose ≤ 111.5 entropy = 0.932 samples = 23 value = [15, 8] class = [
DiabetesPedigreeFunction ≤ 0.49 entropy = 0.332 samples = 49 value = [46, 3] class = [
Age ≤ 27.5 entropy = 0.371 samples = 14 value = [13, 1] class = [
BMI ≤ 32.0 entropy = 0.985 samples = 21 value = [12, 9] class = [
BMI ≤ 24.85 entropy = 0.918 samples = 3 value = [1, 2] class = 1
entropy = 0.0 samples = 4 value = [4, 0] class = [
entropy = 0.0 samples = 2 value = [0, 2] class = 1
DiabetesPedigreeFunction ≤ 0.781 entropy = 0.605 samples = 27 value = [23, 4] class = [
Age ≤ 56.5 entropy = 0.998 samples = 82 value = [43, 39] class = [
DiabetesPedigreeFunction ≤ 0.786 entropy = 0.821 samples = 39 value = [10, 29] class = 1
entropy = 0.0 samples = 2 value = [0, 2] class = 1
DiabetesPedigreeFunction ≤ 0.923 entropy = 0.918 samples = 3 value = [2, 1] class = [
BloodPressure ≤ 61.0 entropy = 0.971 samples = 5 value = [3, 2] class = [
DiabetesPedigreeFunction ≤ 1.308 entropy = 0.592 samples = 21 value = [3, 18] class = 1
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 12 value = [12, 0] class = [
SkinThickness ≤ 40.5 entropy = 0.391 samples = 13 value = [12, 1] class = [
BloodPressure ≤ 72.0 entropy = 0.881 samples = 10 value = [3, 7] class = 1
entropy = 0.0 samples = 32 value = [32, 0] class = [
BloodPressure ≤ 65.0 entropy = 0.672 samples = 17 value = [14, 3] class = [
entropy = 0.0 samples = 12 value = [12, 0] class = [
BloodPressure ≤ 69.5 entropy = 1.0 samples = 2 value = [1, 1] class = [
entropy = 0.0 samples = 3 value = [0, 3] class = 1
BloodPressure ≤ 85.0 entropy = 0.918 samples = 18 value = [12, 6] class = [
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = [
BMI ≤ 32.4 entropy = 0.426 samples = 23 value = [21, 2] class = [
SkinThickness ≤ 35.5 entropy = 1.0 samples = 4 value = [2, 2] class = [
DiabetesPedigreeFunction ≤ 0.442 entropy = 1.0 samples = 77 value = [38, 39] class = 1
entropy = 0.0 samples = 5 value = [5, 0] class = [
Age ≤ 30.0 entropy = 0.439 samples = 22 value = [2, 20] class = 1
Glucose ≤ 150.5 entropy = 0.998 samples = 17 value = [8, 9] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 3 value = [3, 0] class = [
DiabetesPedigreeFunction ≤ 0.142 entropy = 0.31 samples = 18 value = [1, 17] class = 1
Glucose ≤ 178.0 entropy = 0.918 samples = 3 value = [2, 1] class = [
entropy = 0.0 samples = 12 value = [12, 0] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1
BloodPressure ≤ 65.0 entropy = 0.971 samples = 5 value = [3, 2] class = [
entropy = 0.0 samples = 5 value = [0, 5] class = 1
Insulin ≤ 88.5 entropy = 0.918 samples = 9 value = [6, 3] class = [
entropy = 0.0 samples = 8 value = [8, 0] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = [
BloodPressure ≤ 73.0 entropy = 0.811 samples = 16 value = [12, 4] class = [
entropy = 0.0 samples = 2 value = [0, 2] class = 1
Pregnancies ≤ 10.5 entropy = 0.764 samples = 9 value = [7, 2] class = [
entropy = 0.0 samples = 14 value = [14, 0] class = [
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
BloodPressure ≤ 67.0 entropy = 0.993 samples = 69 value = [31, 38] class = 1
SkinThickness ≤ 31.5 entropy = 0.544 samples = 8 value = [7, 1] class = [
Pregnancies ≤ 1.5 entropy = 1.0 samples = 4 value = [2, 2] class = [
entropy = 0.0 samples = 18 value = [0, 18] class = 1
Pregnancies ≤ 3.0 entropy = 0.971 samples = 15 value = [6, 9] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
entropy = 0.0 samples = 1 value = [1, 0] class = [
entropy = 0.0 samples = 17 value = [0, 17] class = 1
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
BMI ≤ 32.6 entropy = 0.918 samples = 3 value = [1, 2] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
entropy = 0.0 samples = 3 value = [3, 0] class = [
BMI ≤ 37.9 entropy = 1.0 samples = 6 value = [3, 3] class = [
Glucose ≤ 147.0 entropy = 0.971 samples = 10 value = [6, 4] class = [
entropy = 0.0 samples = 6 value = [6, 0] class = [
Pregnancies ≤ 4.5 entropy = 0.544 samples = 8 value = [7, 1] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1
BMI ≤ 29.6 entropy = 0.65 samples = 12 value = [2, 10] class = 1
Insulin ≤ 109.0 entropy = 1.0 samples = 57 value = [29, 28] class = [
entropy = 0.0 samples = 7 value = [7, 0] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
entropy = 0.0 samples = 2 value = [0, 2] class = 1
Glucose ≤ 113.5 entropy = 0.954 samples = 8 value = [5, 3] class = [
DiabetesPedigreeFunction ≤ 1.379 entropy = 0.592 samples = 7 value = [1, 6] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = [
entropy = 0.0 samples = 2 value = [0, 2] class = 1
Pregnancies ≤ 3.5 entropy = 0.811 samples = 4 value = [3, 1] class = [
entropy = 0.0 samples = 2 value = [0, 2] class = 1
Insulin ≤ 83.5 entropy = 0.985 samples = 7 value = [3, 4] class = 1
entropy = 0.0 samples = 3 value = [3, 0] class = [
Pregnancies ≤ 2.5 entropy = 1.0 samples = 2 value = [1, 1] class = [
entropy = 0.0 samples = 6 value = [6, 0] class = [
BMI ≤ 27.4 entropy = 0.918 samples = 3 value = [2, 1] class = [
entropy = 0.0 samples = 9 value = [0, 9] class = 1
Pregnancies ≤ 8.5 entropy = 0.968 samples = 43 value = [26, 17] class = [
Age ≤ 40.5 entropy = 0.75 samples = 14 value = [3, 11] class = 1
entropy = 0.0 samples = 3 value = [3, 0] class = [
Age ≤ 37.0 entropy = 0.971 samples = 5 value = [2, 3] class = 1
entropy = 0.0 samples = 6 value = [0, 6] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = [
entropy = 0.0 samples = 3 value = [3, 0] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1
Pregnancies ≤ 1.0 entropy = 0.971 samples = 5 value = [3, 2] class = [
entropy = 0.0 samples = 2 value = [0, 2] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
SkinThickness ≤ 25.0 entropy = 1.0 samples = 34 value = [17, 17] class = [
entropy = 0.0 samples = 9 value = [9, 0] class = [
Glucose ≤ 127.5 entropy = 1.0 samples = 6 value = [3, 3] class = [
entropy = 0.0 samples = 8 value = [0, 8] class = 1
entropy = 0.0 samples = 3 value = [0, 3] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
BMI ≤ 33.75 entropy = 0.918 samples = 3 value = [1, 2] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
BloodPressure ≤ 84.0 entropy = 0.902 samples = 22 value = [7, 15] class = 1
BloodPressure ≤ 95.0 entropy = 0.65 samples = 12 value = [10, 2] class = [
entropy = 0.0 samples = 3 value = [0, 3] class = 1
entropy = 0.0 samples = 3 value = [3, 0] class = [
entropy = 0.0 samples = 1 value = [1, 0] class = [
entropy = 0.0 samples = 2 value = [0, 2] class = 1
BloodPressure ≤ 69.0 entropy = 0.65 samples = 18 value = [3, 15] class = 1
entropy = 0.0 samples = 4 value = [4, 0] class = [
SkinThickness ≤ 40.5 entropy = 0.439 samples = 11 value = [10, 1] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1
entropy = 0.0 samples = 1 value = [1, 0] class = [
Glucose ≤ 126.5 entropy = 0.523 samples = 17 value = [2, 15] class = 1
entropy = 0.0 samples = 8 value = [8, 0] class = [
Insulin ≤ 81.5 entropy = 0.918 samples = 3 value = [2, 1] class = [
entropy = 0.0 samples = 11 value = [0, 11] class = 1
DiabetesPedigreeFunction ≤ 0.251 entropy = 0.918 samples = 6 value = [2, 4] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1
Glucose ≤ 144.0 entropy = 0.918 samples = 3 value = [2, 1] class = [
entropy = 0.0 samples = 3 value = [0, 3] class = 1
entropy = 0.0 samples = 2 value = [2, 0] class = [
entropy = 0.0 samples = 1 value = [0, 1] class = 1