

NAME: GARGI GHOSAL
ROLL: 301911001004
B.E IT 4TH YEAR 1ST SEM
MACHINE LEARNING LAB
ASSIGNMENT #1

UCI datasets (can be loaded from the package itself):

a. Iris plants dataset: <https://archive.ics.uci.edu/ml/datasets/Iris/>

b. Diabetes dataset:

<https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html>

c. Wisconsin Breast Cancer Dataset:

[https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Di](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))
agnostic)

*******SOLUTION*******

1.Employ Naive Bayes (Gaussian, Multinomial & Bernoulli) classifier and show classification results (Accuracy, Precision, Recall, F-score, confusion matrix) with and without parameter tuning.

*******FOR IRISH DATASET*****:**

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
os.chdir(r"C:\Users\ghosa\Projects\ML\ass1\iris")
dataset = pd.read_csv("iris.data")
dataset.head()
```

	s_length(cm)	s_width(cm)	p_length(cm)	p_width(cm)	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

```
dataset.groupby('species').size()
```

```
species
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64
```

```
train, test = train_test_split(dataset, test_size = 0.4)
X_train = train[['s_length(cm)', 's_width(cm)', 'p_length(cm)', 'p_width(cm)']]
y_train = train.species
X_test = test[['s_length(cm)', 's_width(cm)', 'p_length(cm)', 'p_width(cm)']]
y_test = test.species
from sklearn.naive_bayes import GaussianNB
```

#Without Parameter tuning Gaussian NB

```
classifier = GaussianNB().fit(X_train,y_train)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

Confusion Matrix:

```
[[22  0  0]
 [ 0 22  1]
 [ 0  1 14]]
```

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

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

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	22
Iris-versicolor	0.96	0.96	0.96	23
Iris-virginica	0.93	0.93	0.93	15
accuracy			0.97	60
macro avg	0.96	0.96	0.96	60
weighted avg	0.97	0.97	0.97	60

```
iris_gaussian.ipynb
iris_multinomial.ipynb
iris_multinomial.jpg
iris.data
```

```
[21] ✓ 0.7s
... Confusion Matrix:
[[22  0  0]
 [ 0 22  1]
 [ 0  1 14]]

print(classification_report(y_test,y_pred))
[22] ✓ 0.8s
...
      precision    recall  f1-score   support

 Iris-setosa      1.00      1.00      1.00        22
 Iris-versicolor  0.96      0.96      0.96        23
 Iris-virginica   0.93      0.93      0.93        15

 accuracy          0.97          60
 macro avg         0.96      0.96      0.96          60
 weighted avg      0.97      0.97      0.97          60
```

#With Parameter tuning Gaussian NB

```
classifier = GaussianNB(priors=None,var_smoothing=1e-5).fit(X_train,y_train)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

```
print(ConfusionMatrix: )
print(confusion_matrix(y_test,y_pred))
```

Confusion Matrix:

```
[[20  0  0]
 [ 0 17  1]
 [ 0  1 21]]
```

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

[10]

...	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	20
Iris-versicolor	0.94	0.94	0.94	18
Iris-virginica	0.95	0.95	0.95	22
accuracy			0.97	60
macro avg	0.97	0.97	0.97	60
weighted avg	0.97	0.97	0.97	60

```

In [10]: from sklearn.metrics import confusion_matrix
In [11]: cm = confusion_matrix(y_test, y_pred)
In [12]: cm
Out[12]:
array([[19,  0,  0],
       [ 0, 20,  0],
       [ 0,  4, 17]])

In [13]: from sklearn.metrics import classification_report
In [14]: cr = classification_report(y_test, y_pred)
In [15]: cr
Out[15]:
precision    recall  f1-score   support

0.0: 1.00      1.00      1.00      19
1.0: 1.00      1.00      1.00      20
2.0: 0.81      0.88      0.84      17

average / total: 0.94      0.94      0.94      56

```

#Without Parameter tuning Multinomial NB

```

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

```

```

. Confusion Matrix:
[[19  0  0]
 [ 0 20  0]
 [ 0  4 17]]

```

```

print(classification_report(y_test,y_pred))

```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	19
Iris-versicolor	0.83	1.00	0.91	20
Iris-virginica	1.00	0.81	0.89	21
accuracy			0.93	60
macro avg	0.94	0.94	0.93	60
weighted avg	0.94	0.93	0.93	60

```

iris.data
ML1.py
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
[73] ✓ 0.7s
... Confusion Matrix:
[[19  0  0]
 [ 0 20  0]
 [ 0  4 17]]

print(classification_report(y_test,y_pred))
[74] ✓ 0.6s
...
precision    recall  f1-score   support

Iris-setosa      1.00      1.00      1.00        19
Iris-versicolor  0.83      1.00      0.91        20
Iris-virginica   1.00      0.81      0.89        21

accuracy         0.93
macro avg        0.94      0.94      0.93        60
weighted avg     0.94      0.93      0.93        60

```

#With Parameter tuning Multinomial NB

```

from sklearn.naive_bayes import MultinomialNB
classifier=MultinomialNB(alpha=2.5, fit_prior= False, class_prior= None).fit(X_train,y_train)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))

```

```

Confusion Matrix:
[[15  0  0]
 [ 0 24  1]
 [ 0  2 18]]

```

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

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	15
Iris-versicolor	0.92	0.96	0.94	25
Iris-virginica	0.95	0.90	0.92	20
accuracy			0.95	60
macro avg	0.96	0.95	0.95	60
weighted avg	0.95	0.95	0.95	60

```

from sklearn.metrics import classification_report, confusion_matrix

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("Confusion Matrix:")
[[15  0  0]
 [ 0 24  1]
 [ 0  1 20]]

print(classification_report(y_test, y_pred))
...
      precision    recall  f1-score   support

 Iris-setosa       1.00      1.00      1.00        15
 Iris-versicolor   0.92      0.96      0.94        25
 Iris-virginica     0.95      0.90      0.92        20

 accuracy          0.95          0.95          0.95         60
 macro avg         0.96          0.95          0.95         60
 weighted avg      0.95          0.95          0.95         60

```

#Without Parameter tuning Bernoulli NB

```

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

```

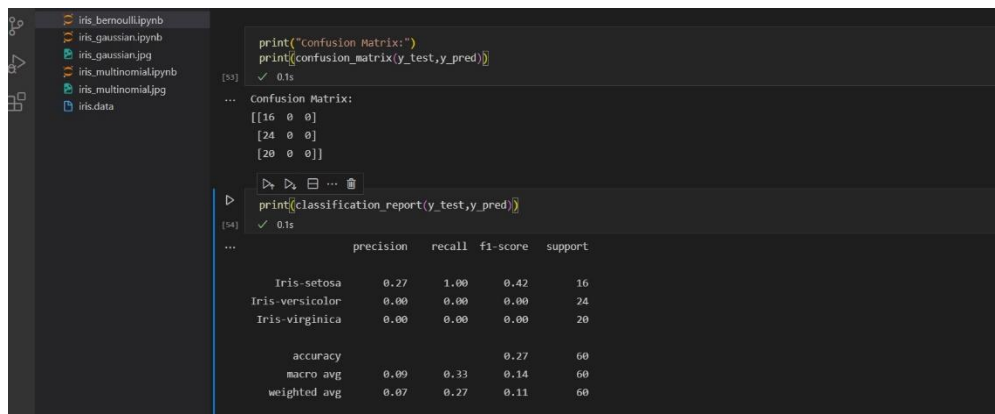
Confusion Matrix:

```
[[ 0  0 21]
 [ 0  0 21]
 [ 0  0 18]]
```

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

	precision	recall	f1-score	support
Iris-setosa	0.00	0.00	0.00	21
Iris-versicolor	0.00	0.00	0.00	21
Iris-virginica	0.30	1.00	0.46	18
accuracy			0.30	60
macro avg	0.10	0.33	0.15	60
weighted avg	0.09	0.30	0.14	60

C:\Users\ghosa\AppData\Local\Programs\Python\Python39\lib\site-pa



```
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))

... Confusion Matrix:
[[16  0  0]
 [24  0  0]
 [20  0  0]]

print(classification_report(y_test,y_pred))

...
              precision    recall  f1-score   support

 Iris-setosa      0.27      1.00      0.42        16
 Iris-versicolor  0.00      0.00      0.00        24
 Iris-virginica   0.00      0.00      0.00        20

 accuracy          0.27          0.30          0.27        60
 macro avg         0.09      0.33      0.14        60
 weighted avg      0.07      0.27      0.11        60
```

#With Parameter tuning Bernoulli NB

```
classifier = BernoulliNB(alpha=1.0, binarize=0, fit_prior=True, class_prior=None).fit(X_train,y_train)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
```



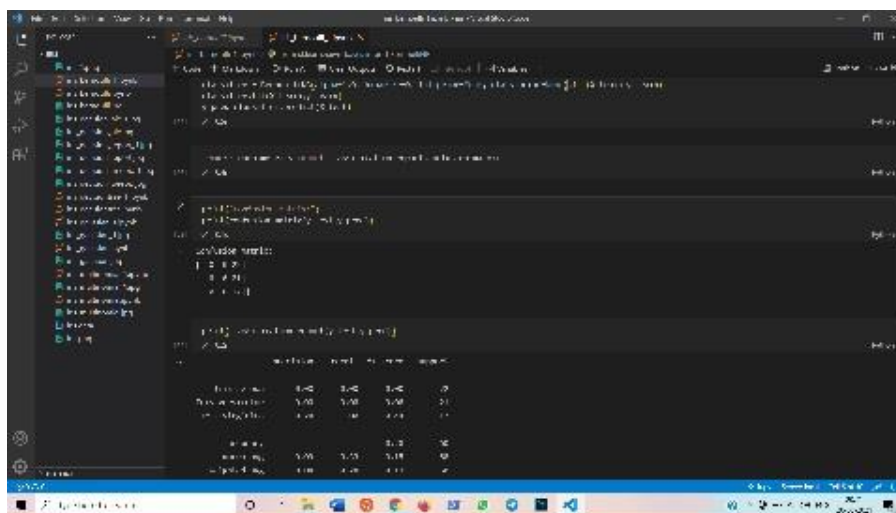
```
print(confusion_matrix(y_test,y_pred))
```

Confusion Matrix:

```
[[ 0  0 22]
 [ 0  0 21]
 [ 0  0 17]]
```

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

	precision	recall	f1-score	support
Iris-setosa	0.00	0.00	0.00	22
Iris-versicolor	0.00	0.00	0.00	21
Iris-virginica	0.28	1.00	0.44	17
accuracy			0.28	60
macro avg	0.09	0.33	0.15	60
weighted avg	0.08	0.28	0.13	60



*****FOR DIABETES DATASET*****.

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
os.chdir(r"C:\Users\ghosa\Projects\ML\ass1\diabetes")
```

```
dataset = pd.read_csv('diabetes.data')
dataset.head()
```

	AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
0	59	2	32.1	101.0	157	93.2	38.0	4.0	4.8598	87	151
1	48	1	21.6	87.0	183	103.2	70.0	3.0	3.8918	69	75
2	72	2	30.5	93.0	156	93.6	41.0	4.0	4.6728	85	141
3	24	1	25.3	84.0	198	131.4	40.0	5.0	4.8903	89	206
4	50	1	23.0	101.0	192	125.4	52.0	4.0	4.2905	80	135

```
X=dataset.drop(['SEX','Y'],axis=1)
y=dataset['SEX']
X_train,X_test,y_train,y_test= train_test_split(X,y, test_size = 0.4)
```

#Without Parameter tuning Gaussian NB

```
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)
from sklearn.metrics import classification_report,confusion_matrix
rint("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

```
Confusion Matrix:
[[32  8]
 [24 25]]
```

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

	precision	recall	f1-score	support
1	0.57	0.80	0.67	40
2	0.76	0.51	0.61	49
accuracy			0.64	89
macro avg	0.66	0.66	0.64	89
weighted avg	0.67	0.64	0.64	89

```

diabetes_gauss.ipynb
diabetes_multi.ipynb
diabetes_multi.jpg
diabetes_report_1.jpg
diabetes_report.jpg
diabetes.data
diabetes.jpg

print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
[33] ✓ 0.1s
... Confusion Matrix:
[[75 29]
 [17 56]]

print(classification_report(y_test,y_pred))
[34] ✓ 0.9s
...
      precision    recall  f1-score   support

     1       0.82     0.72     0.77        104
     2       0.66     0.77     0.71         73

 accuracy          0.74          0.74          0.74        177
 macro avg          0.74          0.74          0.74        177
 weighted avg       0.75          0.74          0.74        177

```

#With Parameter tuning Gaussian NB

```

classifier = GaussianNB(priors=None,var_smoothing=1e-5).fit(X_train,y_train)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
rint("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))

```

```

Confusion Matrix:
[[54 34]
 [28 61]]

```

```

print(classification_report(y_test,y_pred))

```

	precision	recall	f1-score	support
1	0.66	0.61	0.64	88
2	0.64	0.69	0.66	89
accuracy			0.65	177
macro avg	0.65	0.65	0.65	177
weighted avg	0.65	0.65	0.65	177

```

diabetes_decisiontree.jpg
diabetes_gauss_1.ipynb
diabetes_gauss.jpg
diabetes_multi_1.ipynb
diabetes_multi_1.jpg
diabetes_multi.ipynb
diabetes_multi.jpg
diabetes_report_1.jpg
diabetes_report.jpg
diabetes.data
diabetes.jpg

from sklearn.metrics import classification_report, confusion_matrix

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

Confusion Matrix:
[[54 34]
 [28 61]]

print(classification_report(y_test, y_pred))

precision    recall  f1-score   support

     1       0.66      0.61      0.64         88
     2       0.64      0.69      0.66         89

 accuracy          0.65          0.65          0.65         177
 macro avg          0.65          0.65          0.65         177
weighted avg          0.65          0.65          0.65         177

```

#Without Parameter tuning Multinomial NB

```

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

```

```

Confusion Matrix:
[[54 29]
 [30 64]]

```

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

	precision	recall	f1-score	support
1	0.64	0.65	0.65	83
2	0.69	0.68	0.68	94
accuracy			0.67	177
macro avg	0.67	0.67	0.67	177
weighted avg	0.67	0.67	0.67	177

The screenshot shows a Jupyter Notebook with a file explorer on the left containing files like 'diabetes_decisiontree.jpg', 'diabetes_decisiontree.ipynb', 'diabetes_report.jpg', 'diabetes_data', and 'diabetes.jpg'. The main area shows the following code and output:

```
from sklearn.metrics import classification_report, confusion_matrix
```

[7] ✓ 0.4s

```
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

[8] ✓ 0.1s

```
... Confusion Matrix:
[[56 43]
 [21 57]]
```

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

[9] ✓ 0.9s

```
...
      precision    recall  f1-score   support

     1       0.73      0.57      0.64         99
     2       0.57      0.73      0.64         78

 accuracy          0.64         177
 macro avg          0.65      0.65      0.64         177
 weighted avg          0.66      0.64      0.64         177
```

#With Parameter tuning Multinomial NB

```
classifier = MultinomialNB(alpha=2.5, fit_prior= False, class_prior= None).fit
(X_train,y_train)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

```
... Confusion Matrix:
[[54 40]
 [25 58]]
```

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

```
10] ✓ 0.1s
```

```
... precision recall f1-score support
```

1	0.68	0.57	0.62	94
2	0.59	0.70	0.64	83
accuracy			0.63	177
macro avg			0.64	177
weighted avg			0.64	177

```
diabetes_gauss.jpg
diabetes_multi_1.jpg
diabetes_multi_1.jpg
diabetes_multi_1.jpg
diabetes_report_1.jpg
diabetes_report_1.jpg
diabetes.data
diabetes.jpg
```

```
> print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

```
[0] ✓ 0.1s
```

```
... Confusion Matrix:
[[48 38]
 [20 71]]
```

```
[0] ✓ 0.9s
```

```
... precision recall f1-score support
```

1	0.71	0.56	0.62	86
2	0.65	0.78	0.71	91
accuracy			0.67	177
macro avg			0.68	177
weighted avg			0.68	177

#Without Parameter tuning Bernoulli NB

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

Confusion Matrix:

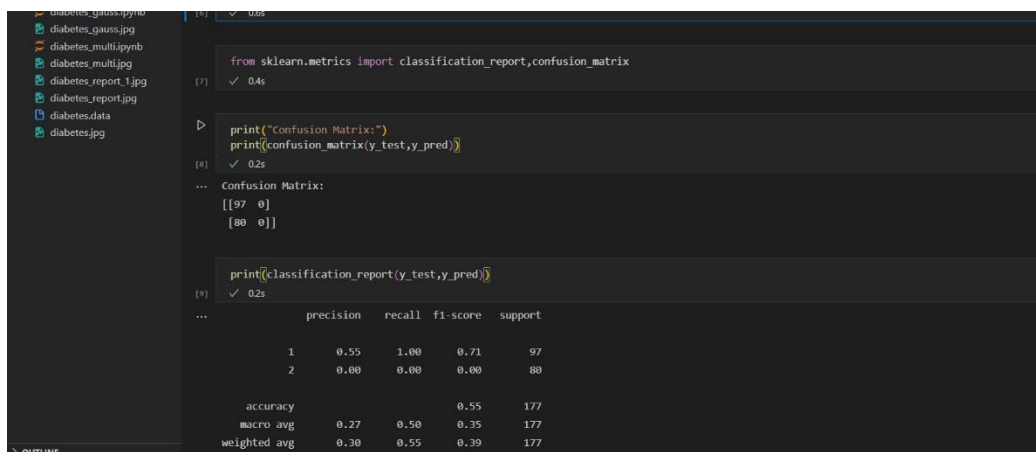
```
[[97  0]
 [80  0]]
```

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

```
..
```

	precision	recall	f1-score	support
1	0.55	1.00	0.71	97
2	0.00	0.00	0.00	80
accuracy			0.55	177
macro avg	0.27	0.50	0.35	177
weighted avg	0.30	0.55	0.39	177

C:\Users\ghosa\AppData\Local\Programs\Python\Python39\lib\site-pack



```
[7] ✓ 0.4s
from sklearn.metrics import classification_report, confusion_matrix

print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))

[8] ✓ 0.2s
...
Confusion Matrix:
[[97  0]
 [80  0]]

[9] ✓ 0.2s
...
precision recall f1-score support
1 0.55 1.00 0.71 97
2 0.00 0.00 0.00 80

accuracy 0.55 177
macro avg 0.27 0.50 0.35 177
weighted avg 0.30 0.55 0.39 177
```

#With Parameter tuning Bernoulli NB

```
classifier = BernoulliNB(alpha=1.0, binarize=0, fit_prior=True, class_prior=None).fit(X_train,y_train)
```

```

classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))

```

```

[24]
... Confusion Matrix:
      [[82  0]
       [95  0]]

```

```

print(classification_report(y_test,y_pred))

```

	precision	recall	f1-score	support
1	0.46	1.00	0.63	82
2	0.00	0.00	0.00	95
accuracy			0.46	177
macro avg	0.23	0.50	0.32	177
weighted avg	0.21	0.46	0.29	177

```

from sklearn.metrics import classification_report,confusion_matrix
[23] ✓ 0.5s

print("confusion Matrix:")
print(confusion_matrix(y_test,y_pred))

[24] ✓ 0.2s
... Confusion Matrix:
      [[82  0]
       [95  0]]

print(classification_report(y_test,y_pred))
[25] ✓ 0.2s
...

```

	precision	recall	f1-score	support
1	0.46	1.00	0.63	82
2	0.00	0.00	0.00	95
accuracy			0.46	177
macro avg	0.23	0.50	0.32	177
weighted avg	0.21	0.46	0.29	177

*****FOR CANCER DATASET*****.

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets
os.chdir(r"C:\Users\ghosa\Projects\ML\ass1\cancer")
dataset= datasets.load_breast_cancer()
print(dataset.feature_names)
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

```
print(dataset.target_names)
```

```
['malignant' 'benign']
```

```
X=dataset.data
y=dataset.target
X_train,X_test,y_train,y_test= train_test_split(X,y, test_size = 0.4)
```

#Without Parameter tuning Gaussian NB

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

```

Confusion Matrix:
[[ 68   8]
 [  4 148]]

```

```

print(classification_report(y_test,y_pred))

```

	precision	recall	f1-score	support
0	0.94	0.89	0.92	76
1	0.95	0.97	0.96	152
accuracy			0.95	228
macro avg	0.95	0.93	0.94	228
weighted avg	0.95	0.95	0.95	228

```

from sklearn.metrics import classification_report, confusion_matrix

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred))

```

	precision	recall	f1-score	support
0	0.94	0.89	0.92	76
1	0.95	0.97	0.96	152
accuracy			0.95	228
macro avg	0.95	0.93	0.94	228
weighted avg	0.95	0.95	0.95	228

#With Parameter tuning Gaussian NB

```

classifier = GaussianNB(priors=None, var_smoothing=1e-5).fit(X_train, y_train)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

```

```
9]
. Confusion Matrix:
[[ 75  18]
 [  1 134]]
```

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

	precision	recall	f1-score	support
0	0.99	0.81	0.89	93
1	0.88	0.99	0.93	135
accuracy			0.92	228
macro avg	0.93	0.90	0.91	228
weighted avg	0.92	0.92	0.91	228

```
[9] from sklearn.metrics import classification_report,confusion_matrix
✓ 0.6s Python

> print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
[10] ✓ 0.6s Python
... Confusion Matrix:
[[ 75  18]
 [  1 134]]

print(classification_report(y_test,y_pred))
[11] ✓ 0.6s Python
...
precision recall f1-score support
0 0.99 0.81 0.89 93
1 0.88 0.99 0.93 135

accuracy
macro avg 0.93 0.90 0.91 228
weighted avg 0.92 0.92 0.91 228
```

#Without Parameter tuning Multinomial NB

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

```
0]
.. Confusion Matrix:
[[ 59  24]
 [  5 140]]
```

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

	precision	recall	f1-score	support
0	0.92	0.71	0.80	83
1	0.85	0.97	0.91	145
accuracy			0.87	228
macro avg	0.89	0.84	0.85	228
weighted avg	0.88	0.87	0.87	228

```
cancer_multi.py
cancer_multi.py:nb
from sklearn.metrics import classification_report,confusion_matrix
[9] ✓ 0.4s Python

print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
[10] ✓ 0.1s Python
... Confusion Matrix:
[[ 59  24]
 [  5 140]]

print(classification_report(y_test,y_pred))
[11] ✓ 0.8s Python
...
precision recall f1-score support
0 0.92 0.71 0.80 83
1 0.85 0.97 0.91 145
accuracy 0.87 228
macro avg 0.89 0.84 0.85 228
weighted avg 0.88 0.87 0.87 228
```

#With Parameter tuning Multinomial NB

```
classifier = MultinomialNB(alpha=2.5, fit_prior= False, class_prior= None).fit(
X_train,y_train)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

```
... Confusion Matrix:
[[ 61  20]
 [   3 144]]
```

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

```
15] ..
```

	precision	recall	f1-score	support
0	0.95	0.75	0.84	81
1	0.88	0.98	0.93	147
accuracy			0.90	228
macro avg	0.92	0.87	0.88	228
weighted avg	0.90	0.90	0.90	228

```
[12] ✓ 0.4s Python
```

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
[13] ✓ 0.3s Python
```

```
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

```
[14] ✓ 0.1s Python
```

```
... Confusion Matrix:
[[ 61  20]
 [   3 144]]
```

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

```
[15] ✓ 0.1s Python
```

```
... precision recall f1-score support
```

0	0.95	0.75	0.84	81
1	0.88	0.98	0.93	147
accuracy			0.90	228
macro avg	0.92	0.87	0.88	228
weighted avg	0.90	0.90	0.90	228

#Without Parameter tuning Bernoulli NB

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

Confusion Matrix:

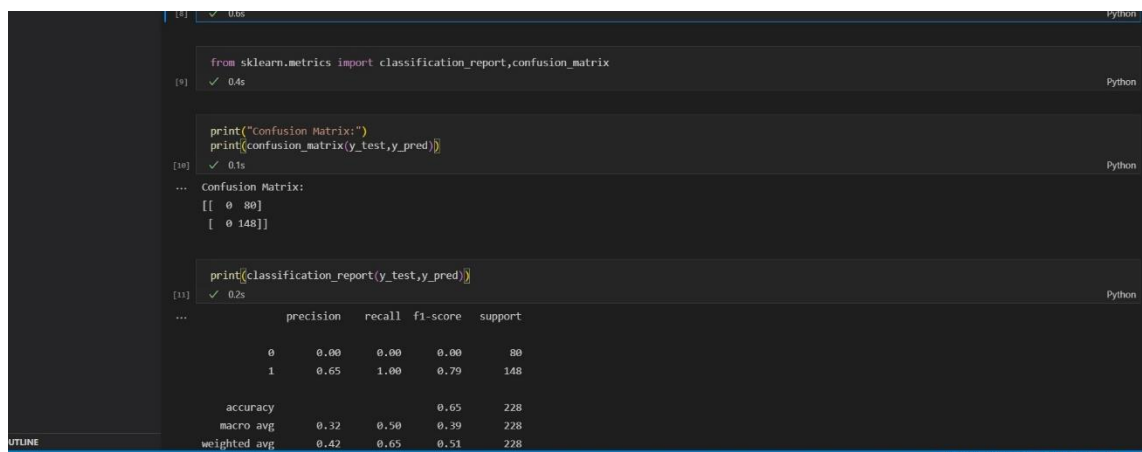
```
[[ 0 86]
 [ 0 142]]
```

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

```
              precision    recall  f1-score   support

     0           0.00        0.00        0.00         86
     1           0.62        1.00        0.77        142

 accuracy              0.62         228
 macro avg           0.31         0.50         0.38         228
 weighted avg        0.39         0.62         0.48         228
```



```
[9] ✓ 0.0s Python
from sklearn.metrics import classification_report, confusion_matrix

[9] ✓ 0.4s Python

[10] ✓ 0.1s Python
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

... Confusion Matrix:
[[ 0 80]
 [ 0 148]]

[11] ✓ 0.2s Python
print(classification_report(y_test, y_pred))

...              precision    recall  f1-score   support

     0           0.00        0.00        0.00         80
     1           0.65        1.00        0.79        148

 accuracy              0.65         228
 macro avg           0.32         0.50         0.39         228
 weighted avg        0.42         0.65         0.51         228
```

#With Parameter tuning Bernoulli NB

```
classifier = BernoulliNB(alpha=1.0, binarize=0, fit_prior=True, class_prior=None).fit(X_train, y_train)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```

10]
.. Confusion Matrix:
   [[ 0  93]
    [ 0 135]]

```

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

```

              precision    recall  f1-score   support

     0       0.00         0.00         0.00         93
     1       0.59         1.00         0.74        135

 accuracy          0.59         228
 macro avg       0.30         0.50         0.37         228
 weighted avg    0.35         0.59         0.44         228

```

```

from sklearn.metrics import classification_report, confusion_matrix

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred))

```

Confusion Matrix:
 [[0 93]
 [0 135]]

```

              precision    recall  f1-score   support

     0       0.00         0.00         0.00         93
     1       0.59         1.00         0.74        135

 accuracy          0.59         228
 macro avg       0.30         0.50         0.37         228
 weighted avg    0.35         0.59         0.44         228

```

C:\Users\ghosa\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\metrics_classification.py:1248: UndefinedMetricWarning: Precision and F-

2. Use Decision Tree classifier for all the three datasets and show classification results (Accuracy, Precision, Recall, F-score, confusion matrix) with and without parameter tuning. Generate the decision tree images for all cases highlighting information like Gini and Entropy.

*****FOR IRISH DATASET*****.

```
train, test = train_test_split(dataset, test_size = 0.4)
X_train = train[['s_length(cm)', 's_width(cm)', 'p_length(cm)', 'p_width(cm)']]
y_train = train.species
X_test = test[['s_length(cm)', 's_width(cm)', 'p_length(cm)', 'p_width(cm)']]
y_test = test.species
from sklearn.tree import DecisionTreeClassifier
```

#Without Parameter tuning Decision Tree Classifier (by default Gini)

```
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Confusion Matrix:

```
[[20  0  0]
 [ 0 21  0]
 [ 0  3 16]]
```

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

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	20
Iris-versicolor	0.88	1.00	0.93	21
Iris-virginica	1.00	0.84	0.91	19
accuracy			0.95	60
macro avg	0.96	0.95	0.95	60
weighted avg	0.96	0.95	0.95	60

#With Parameter tuning Decision Tree Classifier (Entropy)

```
classifier = DecisionTreeClassifier(criterion="entropy",max_depth=20)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

Confusion Matrix:

```
[[19  0  0]
 [ 0 19  2]
 [ 0  3 17]]
```

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

..	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	19
Iris-versicolor	0.86	0.90	0.88	21
Iris-virginica	0.89	0.85	0.87	20
accuracy			0.92	60
macro avg	0.92	0.92	0.92	60
weighted avg	0.92	0.92	0.92	60

```
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
[9] ✓ 0.1s
... Confusion Matrix:
[[19  0  0]
 [ 0 19  2]
 [ 0  3 17]]

> print(classification_report(y_test,y_pred))
[10] ✓ 0.1s
... precision recall f1-score support

Iris-setosa      1.00      1.00      1.00      19
Iris-versicolor  0.86      0.90      0.88      21
Iris-virginica   0.89      0.85      0.87      20

accuracy         0.92      60
macro avg        0.92      0.92      0.92      60
weighted avg     0.92      0.92      0.92      60
```

```
from sklearn import tree
text_representation = tree.export_text(classifier)
print(text_representation)
```

```

iris_decisiontree_1.py
iris_decisiontree.ipynb
iris_gaussian.ipynb
iris_gaussian.jpg
iris_multinomial.ipynb
iris_multinomial.jpg
iris.data
iris.png

from sklearn import tree
text_representation = tree.export_text(classifier)
print(text_representation)

```

```

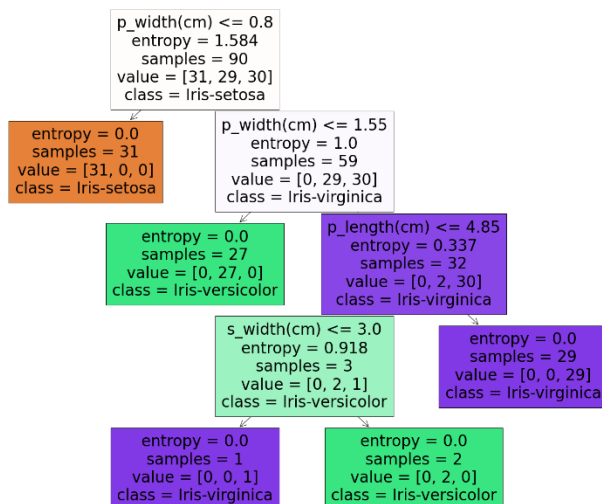
... |--- feature_3 <= 0.80
    | |--- class: Iris-setosa
    |--- feature_3 > 0.80
    | |--- feature_3 <= 1.55
    | | |--- class: Iris-versicolor
    | |--- feature_3 > 1.55
    | | |--- feature_2 <= 4.85
    | | | |--- feature_1 <= 3.00
    | | | | |--- class: Iris-virginica
    | | | |--- feature_1 > 3.00
    | | | | |--- class: Iris-versicolor
    | | |--- feature_2 > 4.85
    | | | |--- class: Iris-virginica

```

```

from sklearn.tree import plot_tree
fig=plt.figure(figsize=(25,20))
fig= plot_tree(classifier,
               feature_names = ["s_length(cm)", "s_width(cm)", "p_length(cm)", "p_width(cm)"],
               class_names = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], filled = True) #
plt.savefig("iris_1.png")

```



*****FOR DIABETES DATASET*****:

#Without Parameter tuning Decision Tree Classifier (by default Gini)

```

X=dataset.drop(['SEX','Y'],axis=1)
y=dataset['SEX']
X_train,X_test,y_train,y_test= train_test_split(X,y, test_size = 0.4)
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix

print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))

```

```
... Confusion Matrix:
[[54 38]
 [44 41]]
```

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

	precision	recall	f1-score	support
1	0.55	0.59	0.57	92
2	0.52	0.48	0.50	85
accuracy			0.54	177
macro avg	0.54	0.53	0.53	177
weighted avg	0.54	0.54	0.54	177

The screenshot shows a Jupyter Notebook with a file explorer on the left containing 'diabetes_decisiontree.ipynb', 'diabetes_decisiontree.jpg', 'diabetes.data', and 'diabetes.jpg'. The main area shows two code cells. The first cell prints the confusion matrix, and the second cell prints the classification report. The outputs are displayed below each code cell.

```
[12]: print("Confusion Matrix:")
      print(confusion_matrix(y_test,y_pred))
... Confusion Matrix:
[[54 38]
 [44 41]]
```

```
[13]: print(classification_report(y_test,y_pred))
... precision recall f1-score support
1      0.55    0.59    0.57     92
2      0.52    0.48    0.50     85

accuracy      0.54
macro avg     0.54    0.53    0.53
weighted avg  0.54    0.54    0.54
```

```
from sklearn import tree
text_representation = tree.export_text(classifier)
print(text_representation)
```



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

	precision	recall	f1-score	support
1	0.61	0.60	0.60	92
2	0.57	0.59	0.58	85
accuracy			0.59	177
macro avg	0.59	0.59	0.59	177
weighted avg	0.59	0.59	0.59	177

The screenshot shows a Jupyter Notebook with the following code and output:

```
from sklearn.metrics import classification_report, confusion_matrix
```

[16] ✓ 0.4s

```
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

[17] ✓ 0.1s

```
Confusion Matrix:
[[55 37]
 [35 50]]
```

[18] ✓ 0.1s

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

[18] ✓ 0.1s

```
precision recall f1-score support
```

1	0.61	0.60	0.60	92
2	0.57	0.59	0.58	85
accuracy			0.59	177
macro avg	0.59	0.59	0.59	177
weighted avg	0.59	0.59	0.59	177

```
from sklearn import tree
text_representation = tree.export_text(classifier)
print(text_representation)
```

The screenshot shows the output of the `tree.export_text` function in a Visual Studio Code window. The output is a text-based representation of the decision tree structure, showing splits on features and resulting classes.

```
from sklearn import tree
text_representation = tree.export_text(classifier)
print(text_representation)
```

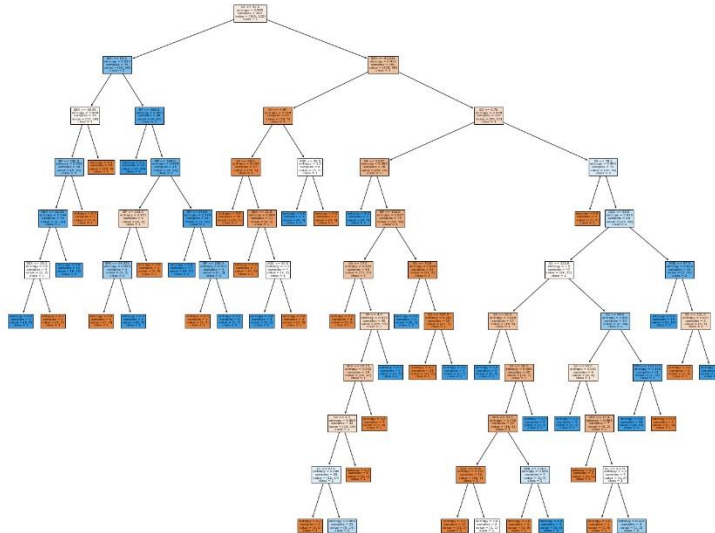
```

--- feature_5 <= 41.50
| --- feature_2 <= 92.00
| | --- feature_3 <= 26.75
| | | --- feature_3 <= 196.00
| | | | --- feature_1 <= 22.85
| | | | | --- feature_3 <= 20.20
| | | | | | --- class: 2
| | | | | | --- feature_1 >= 20.70
| | | | | | --- class: 1
| | | | | --- feature_1 >= 22.85
| | | | | --- class: 2
| | | | | --- feature_1 >= 196.00
| | | | | --- class: 1
| | | | | --- feature_1 >= 20.35
| | | | | --- class: 1
| | | | | --- feature_2 >= 92.00
| | | | | --- feature_3 <= 188.50
| | | | | --- class: 2
| | | | | --- feature_4 >= 188.50
| | | | | --- feature_3 <= 199.00
| | | | | --- feature_3 <= 192.50
| | | | | | --- feature_2 <= 94.82
| | | | | | --- class: 1
| | | | | | --- feature_2 >= 94.81
| | | | | | --- class: 2

```

```
from sklearn.tree import plot_tree
fig=plt.figure(figsize=(25,20))
```

```
fig=tree.plot_tree(decision_tree=classifier,feature_names=dataset.columns,clas
s_names=['1','2'],filled=True)
plt.savefig("diabetes_1.jpg")
```



*******FOR CANCER DATASET*****.**

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import datasets
os.chdir(r"C:\Users\ghosa\Projects\ML\ass1\cancer")
dataset= datasets.load_breast_cancer()
print(dataset.feature_names)
```

```
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

```
print(dataset.target_names)
['malignant' 'benign']
```

```
X=dataset.data
y=dataset.target
X_train,X_test,y_train,y_test= train_test_split(X,y, test_size = 0.4)
from sklearn.tree import DecisionTreeClassifier
```

#Without Parameter tuning Decision Tree Classifier (by default Gini)

```
classifier = DecisionTreeClassifier()
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

```
.. Confusion Matrix:
[[ 71   8]
 [  5 144]]
```

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


	precision	recall	f1-score	support
0	0.93	0.90	0.92	79
1	0.95	0.97	0.96	149
accuracy			0.94	228
macro avg	0.94	0.93	0.94	228
weighted avg	0.94	0.94	0.94	228

```

cancer_ben.jpg
cancer_dec.ipynb
cancer_gauss_1.jpg
cancer_gauss.ipynb
cancer_gauss.jpg
cancer_multi_1.jpg
cancer_multi.ipynb
cancer_multi.jpg
cancer.jpg

from sklearn.metrics import classification_report, confusion_matrix

[9] ✓ 0.4s

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

[10] ✓ 0.1s
...
Confusion Matrix:
[[ 71  8]
 [ 5 144]]

print(classification_report(y_test, y_pred))

[11] ✓ 0.1s
...
      precision    recall  f1-score   support

      0       0.93       0.90       0.92         79
      1       0.95       0.97       0.96        149

   accuracy          0.94          0.93          0.94        228
  macro avg       0.94       0.93       0.94        228
 weighted avg       0.94       0.94       0.94        228

from sklearn import tree

```

```

from sklearn import tree
text_representation = tree.export_text(classifier)
print(text_representation)

```

```

cancer_ben.jpg
cancer_dec.ipynb
cancer_gauss_1.jpg
cancer_gauss.ipynb
cancer_gauss.jpg
cancer_multi_1.jpg
cancer_multi.ipynb
cancer_multi.jpg
cancer.jpg

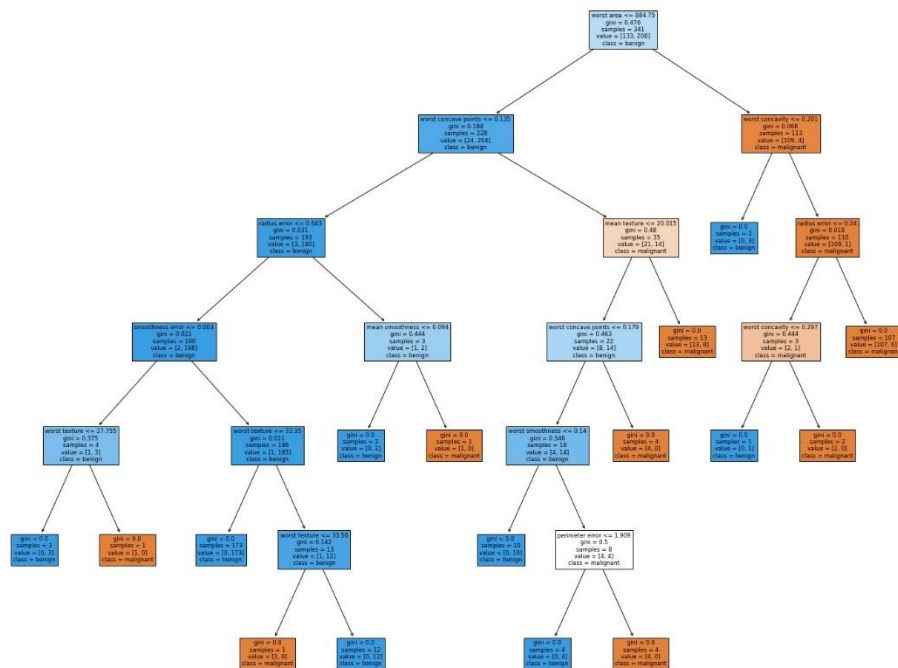
from sklearn import tree
text_representation = tree.export_text(classifier)
print(text_representation)

[12] ✓ 0.1s
...
|--- feature_23 <= 884.75
|   |--- feature_27 <= 0.13
|   |   |--- feature_10 <= 0.64
|   |   |   |--- feature_14 <= 0.00
|   |   |   |   |--- feature_21 <= 27.76
|   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_21 > 27.76
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_14 > 0.00
|   |   |   |   |   |   |   |--- feature_21 <= 33.35
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |--- feature_21 > 33.35
|   |   |   |   |   |   |   |   |   |--- feature_21 <= 33.56
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- feature_21 > 33.56
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |   |--- feature_10 > 0.64
|   |   |   |   |   |   |   |   |   |   |   |   |--- feature_4 <= 0.09
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |   |   |   |--- feature_4 > 0.09
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |--- feature_27 > 0.13
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |--- feature_1 <= 20.01
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |--- feature_27 <= 0.18
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |--- feature_24 <= 0.14

```

```
from sklearn.tree import plot_tree
```

```
fig=plt.figure(figsize=(25,20))
fig=tree.plot_tree(decision_tree=classifier,feature_names=dataset.feature_names,class_names=['malignant','benign'],filled=True)
plt.savefig("cancer.jpg")
```



#With Parameter tuning Decision Tree Classifier (Entropy)

```
classifier = DecisionTreeClassifier(criterion="entropy",max_depth=10)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test,y_pred))
```

```
Confusion Matrix:
[[ 79   6]
 [  6 137]]
```

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

	precision	recall	f1-score	support
0	0.93	0.93	0.93	85
1	0.96	0.96	0.96	143
accuracy			0.95	228
macro avg	0.94	0.94	0.94	228
weighted avg	0.95	0.95	0.95	228

```

from sklearn.metrics import classification_report, confusion_matrix

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred))

```

Confusion Matrix:

```

[[ 82  9]
 [ 8 129]]

```

Classification Report:

```

              precision    recall  f1-score   support

    0       0.91      0.90      0.91         91
    1       0.93      0.94      0.94        137

 accuracy          0.92      0.92      0.92        228
 macro avg          0.92      0.92      0.92        228
 weighted avg          0.93      0.93      0.93        228

```

```

from sklearn import tree
text_representation = tree.export_text(classifier)
print(text_representation)

```

```

from sklearn import tree
text_representation = tree.export_text(classifier)
print(text_representation)

```

```

--- feature_22 <= 116.80
|   --- feature_27 <= 0.13
|   |   --- feature_13 <= 42.19
|   |   |   --- feature_19 <= 0.00
|   |   |   |   --- feature_27 <= 0.10
|   |   |   |   |   --- class: 1
|   |   |   |   |   --- feature_27 > 0.10
|   |   |   |   |   |   --- class: 0
|   |   |   |   |   |   --- feature_19 > 0.00
|   |   |   |   |   |   |   --- class: 1
|   |   |   |   |   |   |   --- feature_13 > 42.19
|   |   |   |   |   |   |   |   --- feature_20 <= 16.88
|   |   |   |   |   |   |   |   |   --- feature_5 <= 0.06
|   |   |   |   |   |   |   |   |   |   --- class: 0
|   |   |   |   |   |   |   |   |   |   --- feature_5 > 0.06
|   |   |   |   |   |   |   |   |   |   |   --- class: 1
|   |   |   |   |   |   |   |   |   |   |   --- feature_20 > 16.88
|   |   |   |   |   |   |   |   |   |   |   |   --- class: 0
|   |   |   |   |   |   |   |   |   |   |   |   --- feature_27 > 0.13
|   |   |   |   |   |   |   |   |   |   |   |   |   --- feature_24 <= 0.13
|   |   |   |   |   |   |   |   |   |   |   |   |   |   --- class: 1
|   |   |   |   |   |   |   |   |   |   |   |   |   |   --- feature_24 > 0.13
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   --- feature_22 <= 102.05
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   --- feature_29 <= 0.10
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   --- class: 1

```

```

from sklearn.tree import plot_tree
fig=plt.figure(figsize=(25,20))
fig=tree.plot_tree(decision_tree=classifier,feature_names=dataset.feature_names,class_names=['malignant','benign'],filled=True)
plt.savefig("cancer_1.jpg")

```

