AS6PR1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

data = pd.read_csv("Breast_Cancer_Dataset.csv")
data.head()
```

```
         id diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
0    842302         M        17.99         10.38          122.80
1001.0
1    842517         M        20.57         17.77          132.90
1326.0
2  84300903         M        19.69         21.25          130.00
1203.0
3  84348301         M        11.42         20.38           77.58
386.1
4  84358402         M        20.29         14.34          135.10
1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
0.10520
4          0.10030           0.13280          0.1980
0.10430

   ...  texture_worst  perimeter_worst  area_worst
smoothness_worst  \
0  ...          17.33           184.60      2019.0            0.1622
```

```
1  ...          23.41           158.80      1956.0            0.1238

2  ...          25.53           152.50      1709.0            0.1444

3  ...          26.50            98.87       567.7            0.2098

4  ...          16.67           152.20      1575.0            0.1374


   compactness_worst  concavity_worst  concave points_worst
symmetry_worst  \
0             0.6656           0.7119                0.2654
0.4601
1             0.1866           0.2416                0.1860
0.2750
2             0.4245           0.4504                0.2430
0.3613
3             0.8663           0.6869                0.2575
0.6638
4             0.2050           0.4000                0.1625
0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]

data.isna().sum()

id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
```

```
compactness_se               0
concavity_se                 0
concave points_se            0
symmetry_se                  0
fractal_dimension_se         0
radius_worst                 0
texture_worst                0
perimeter_worst              0
area_worst                   0
smoothness_worst             0
compactness_worst            0
concavity_worst              0
concave points_worst         0
symmetry_worst               0
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

```
data = data.drop(['Unnamed: 32'], axis=1)
```

```
le=LabelEncoder()
```

```
data["diagnosis"]= le.fit_transform(data["diagnosis"])
```

```
le.classes_
```

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

```
data.head()
```

```
         id  diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
0    842302          1        17.99         10.38          122.80
1001.0
1    842517          1        20.57         17.77          132.90
1326.0
2  84300903          1        19.69         21.25          130.00
1203.0
3  84348301          1        11.42         20.38           77.58
386.1
4  84358402          1        20.29         14.34          135.10
1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
```

```
0.10520
4          0.10030           0.13280          0.1980
0.10430

   ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  ...         25.38          17.33           184.60      2019.0
1  ...         24.99          23.41           158.80      1956.0
2  ...         23.57          25.53           152.50      1709.0
3  ...         14.91          26.50            98.87       567.7
4  ...         22.54          16.67           152.20      1575.0

   smoothness_worst  compactness_worst  concavity_worst  concave
points_worst  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758
3          0.6638                  0.17300
4          0.2364                  0.07678

[5 rows x 32 columns]
```

```
X = data.drop(columns = "diagnosis")
Y = data["diagnosis"]
```

```
X.head()
```

```
         id  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302        17.99         10.38          122.80     1001.0
1    842517        20.57         17.77          132.90     1326.0
2  84300903        19.69         21.25          130.00     1203.0
3  84348301        11.42         20.38           77.58      386.1
4  84358402        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
```

```
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
0.10520
4          0.10030           0.13280          0.1980
0.10430

   symmetry_mean  ...  radius_worst  texture_worst  perimeter_worst  \
0         0.2419  ...         25.38          17.33           184.60
1         0.1812  ...         24.99          23.41           158.80
2         0.2069  ...         23.57          25.53           152.50
3         0.2597  ...         14.91          26.50            98.87
4         0.1809  ...         22.54          16.67           152.20

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   concave points_worst  symmetry_worst  fractal_dimension_worst
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 31 columns]

Y.head()

0    1
1    1
2    1
3    1
4    1
Name: diagnosis, dtype: int64

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size =
0.3,random_state = 1)

reg = LogisticRegression()
reg.fit(X_train,Y_train)

LogisticRegression()

Y_pred = reg.predict(X_test)
```

```
reg_acc = accuracy_score(Y_test,Y_pred)
print("Logisticregressionaccurace=",reg_acc*100)
```

```
Logisticregressionaccurace= 94.15204678362574
```

```
gaussian_nb=GaussianNB()
gaussian_nb.fit(X_train,Y_train)
```

```
GaussianNB()
```

```
Y_pred = gaussian_nb.predict(X_test)
```

```
bayes_acc = accuracy_score(Y_test,Y_pred)
print("Naive bayes accuracy=",bayes_acc*100)
```

```
Naive bayes accuracy= 94.15204678362574
```

```
knn5=KNeighborsClassifier(n_neighbors=5)
knn5.fit(X_train,Y_train)
```

```
KNeighborsClassifier()
```

```
Y_pred=knn5.predict(X_test)
```

```
KNN_acc = accuracy_score(Y_test,Y_pred)
KNN_acc*100
```

```
94.15204678362574
```

```
svc = SVC(C=3)
svc.fit(X_train,Y_train)
```

```
SVC(C=3)
```

```
Y_pred=svc.predict(X_test)
svc_acc=accuracy_score(Y_test,Y_pred)
svc_acc*100
```

```
63.1578947368421
```

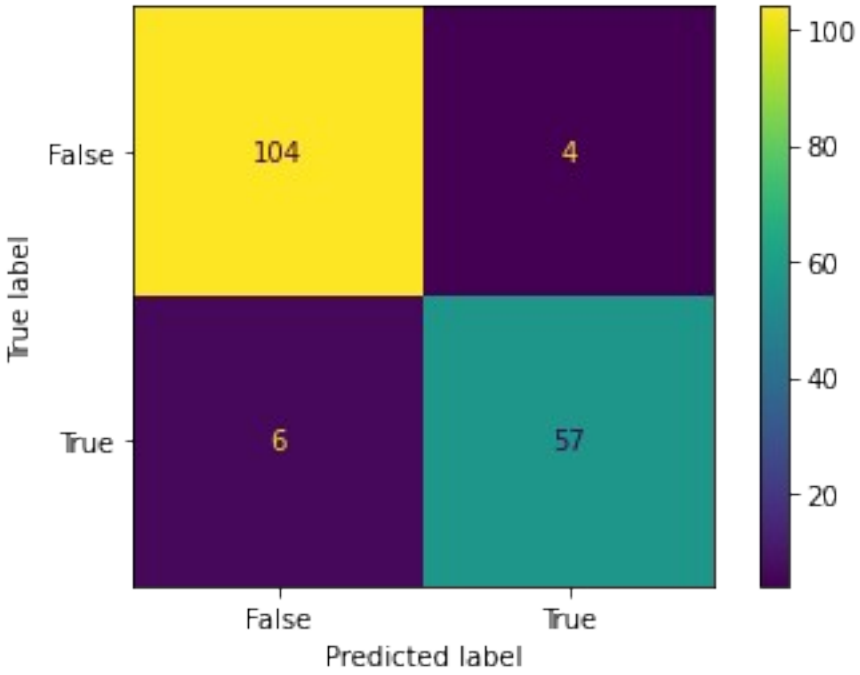
```
dt=DecisionTreeClassifier(criterion='entropy',splitter='best',max_feat
ures=3,max_depth=3)
dt.fit(X_train,Y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3,
max_features=3)
```

```
Y_pred = dt.predict(X_test)
DT_acc = accuracy_score(Y_test,Y_pred)
print("Decision tree accuracy=",DT_acc*100)
```

```
Decision tree accuracy= 87.13450292397661
```

```
RF=RandomForestClassifier(n_estimators=100,
criterion='entropy',max_depth=3)
RF.fit(X_train,Y_train)

RandomForestClassifier(criterion='entropy', max_depth=3)

Y_pred = RF.predict(X_test)
RF_acc = accuracy_score(Y_test,Y_pred)
print("Random forest accuracy=",RF_acc*100)

Random forest accuracy= 94.15204678362574

con_mat = metrics.confusion_matrix(Y_test,Y_pred)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
con_mat, display_labels = [False, True])

cm_display.plot()
plt.show()
```



AS6PR2

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import  classification_report


data = pd.read_csv("Breast_Cancer_Dataset.csv")



data.isna().sum()


id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave points_mean         0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
```

```
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
fractal_dimension_worst     0
Unnamed: 32               569
dtype: int64
```

```
data= data.drop(['Unnamed: 32'], axis=1)
le = LabelEncoder()
data["diagnosis"]= le.fit_transform(data["diagnosis"])
data
```

```
           id  diagnosis  radius_mean  texture_mean  perimeter_mean  \
0      842302          1        17.99         10.38          122.80
1      842517          1        20.57         17.77          132.90
2    84300903          1        19.69         21.25          130.00
3    84348301          1        11.42         20.38           77.58
4    84358402          1        20.29         14.34          135.10
..        ...        ...          ...           ...             ...
564    926424          1        21.56         22.39          142.00
565    926682          1        20.13         28.25          131.20
566    926954          1        16.60         28.08          108.30
567    927241          1        20.60         29.33          140.10
568     92751          0         7.76         24.54           47.92

     area_mean  smoothness_mean  compactness_mean  concavity_mean  \
0       1001.0          0.11840           0.27760         0.30010
1       1326.0          0.08474           0.07864         0.08690
2       1203.0          0.10960           0.15990         0.19740
3        386.1          0.14250           0.28390         0.24140
4       1297.0          0.10030           0.13280         0.19800
..         ...              ...               ...             ...
564     1479.0          0.11100           0.11590         0.24390
565     1261.0          0.09780           0.10340         0.14400
566      858.1          0.08455           0.10230         0.09251
567     1265.0          0.11780           0.27700         0.35140
568      181.0          0.05263           0.04362         0.00000

     concave points_mean  ...  radius_worst  texture_worst
perimeter_worst  \
0                0.14710  ...        25.380          17.33
184.60
```

```
1               0.07017  ...        24.990          23.41
158.80
2               0.12790  ...        23.570          25.53
152.50
3               0.10520  ...        14.910          26.50
98.87
4               0.10430  ...        22.540          16.67
152.20
..                  ...  ...           ...            ...
...
564             0.13890  ...        25.450          26.40
166.10
565             0.09791  ...        23.690          38.25
155.00
566             0.05302  ...        18.980          34.12
126.70
567             0.15200  ...        25.740          39.42
184.60
568             0.00000  ...         9.456          30.37
59.16

     area_worst  smoothness_worst  compactness_worst  concavity_worst
\
0        2019.0           0.16220            0.66560           0.7119

1        1956.0           0.12380            0.18660           0.2416

2        1709.0           0.14440            0.42450           0.4504

3         567.7           0.20980            0.86630           0.6869

4        1575.0           0.13740            0.20500           0.4000

..          ...               ...                ...              ...

564      2027.0           0.14100            0.21130           0.4107

565      1731.0           0.11660            0.19220           0.3215

566      1124.0           0.11390            0.30940           0.3403

567      1821.0           0.16500            0.86810           0.9387

568       268.6           0.08996            0.06444           0.0000


     concave points_worst  symmetry_worst  fractal_dimension_worst
0                  0.2654          0.4601                  0.11890
1                  0.1860          0.2750                  0.08902
```

```
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678
..                  ...             ...                      ...
564              0.2216          0.2060                  0.07115
565              0.1628          0.2572                  0.06637
566              0.1418          0.2218                  0.07820
567              0.2650          0.4087                  0.12400
568              0.0000          0.2871                  0.07039

[569 rows x 32 columns]
```

```
X = data.drop(columns = "diagnosis")
Y = data["diagnosis"]

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size =
0.3,random_state = 1)

reg = LogisticRegression()
reg.fit(X_train,Y_train)
Y_pred = reg.predict(X_test)
reg_acc = accuracy_score(Y_test,Y_pred)

gaussian_nb=GaussianNB()
gaussian_nb.fit(X_train,Y_train)
Y_pred = gaussian_nb.predict(X_test)
bayes_acc = accuracy_score(Y_test,Y_pred)

knn5=KNeighborsClassifier(n_neighbors=5)
knn5.fit(X_train,Y_train)
Y_pred=knn5.predict(X_test)
KNN_acc = accuracy_score(Y_test,Y_pred)

svc = SVC(C=10)
svc.fit(X_train,Y_train)
Y_pred=svc.predict(X_test)
svc_acc=accuracy_score(Y_test,Y_pred)

dt=DecisionTreeClassifier(criterion='entropy',splitter='best',max_feat
ures=3,max_depth=3)
dt.fit(X_train,Y_train)
Y_pred = dt.predict(X_test)
DT_acc = accuracy_score(Y_test,Y_pred)

RF=RandomForestClassifier(n_estimators=100,
criterion='entropy',max_depth=3)
RF.fit(X_train,Y_train)
Y_pred = RF.predict(X_test)
RF_acc = accuracy_score(Y_test,Y_pred)

print("LogisticRegression -> Accuracy -> ",reg_acc)
print("Naive_Bayes -> Accuracy -> ",bayes_acc)
```

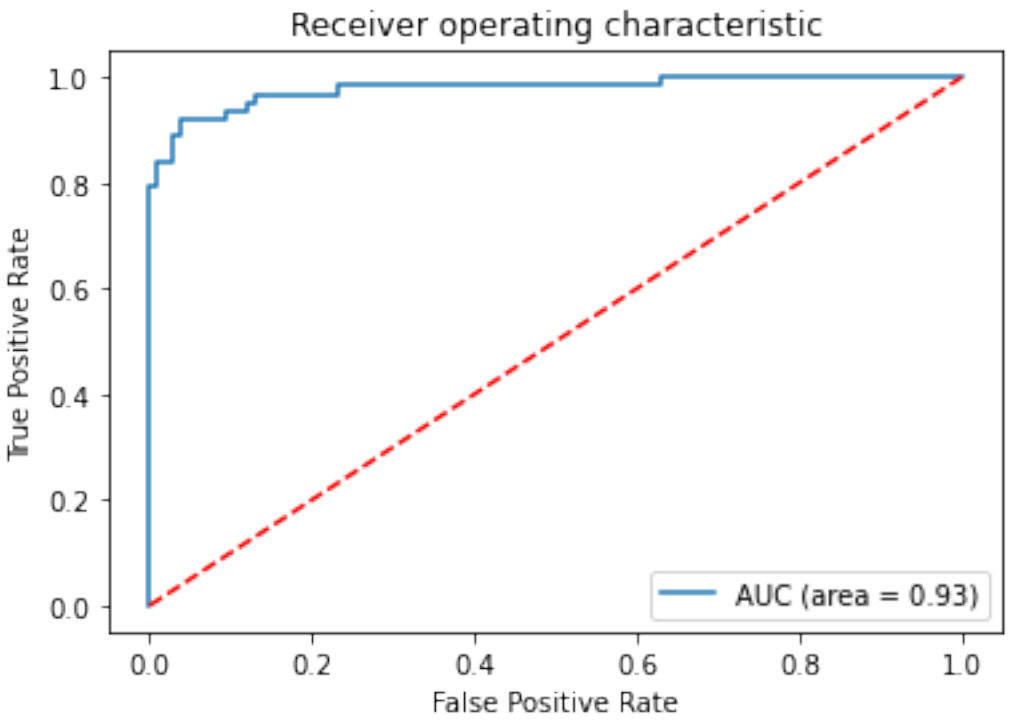
```
print("Decision_Tree -> Accuracy -> ",DT_acc)
print("K-NN -> Accuracy -> ",KNN_acc)
print("SVM -> Accuracy -> ",svc_acc)
print("Random_Forest -> Accuracy -> ",RF_acc)
```

```
LogisticRegression -> Accuracy ->  0.631578947368421
Naive_Bayes -> Accuracy ->  0.631578947368421
Decision_Tree -> Accuracy ->  0.9064327485380117
K-NN -> Accuracy ->  0.7660818713450293
SVM -> Accuracy ->  0.631578947368421
Random_Forest -> Accuracy ->  0.935672514619883
```

```
RF_roc_auc =metrics.roc_auc_score(Y_test, RF.predict(X_test))

print ("ROC_AUC Score is -> ", RF_roc_auc )
```

```
ROC_AUC Score is ->  0.925925925925926
```

```
lfpr, ltpr, lthresholds = metrics.roc_curve(Y_test,
RF.predict_proba(X_test)[:,1])
plt.plot(lfpr, ltpr, label='AUC (area = %0.2f)' % RF_roc_auc)

plt.plot([0, 1], [0, 1],'r--')
#plt.xlim([0.0, 1.0])
#plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```




```
print("F1 Score is : ",f1_score(Y_test, Y_pred))
```

```
F1 Score is :  0.9105691056910569
```

```
print(classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       108
           1       0.93      0.89      0.91        63

    accuracy                           0.94       171
   macro avg       0.94      0.93      0.93       171
weighted avg       0.94      0.94      0.94       171
```

AS6PR3

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import  classification_report
from sklearn.model_selection import GridSearchCV

data = pd.read_csv("voice.csv")
data.head()
```

```
   meanfreq        sd    median       Q25       Q75       IQR
skew  \
0  0.059781  0.064241  0.032027  0.015071  0.090193  0.075122
12.863462
1  0.066009  0.067310  0.040229  0.019414  0.092666  0.073252
22.423285
2  0.077316  0.083829  0.036718  0.008701  0.131908  0.123207
30.757155
3  0.151228  0.072111  0.158011  0.096582  0.207955  0.111374
1.232831
4  0.135120  0.079146  0.124656  0.078720  0.206045  0.127325
1.101174

          kurt    sp.ent       sfm  ...  centroid   meanfun    minfun
\
0   274.402906  0.893369  0.491918  ...  0.059781  0.084279  0.015702

1   634.613855  0.892193  0.513724  ...  0.066009  0.107937  0.015826

2  1024.927705  0.846389  0.478905  ...  0.077316  0.098706  0.015656

3     4.177296  0.963322  0.727232  ...  0.151228  0.088965  0.017798

4     4.333713  0.971955  0.783568  ...  0.135120  0.106398  0.016931
```

```
     maxfun   meandom    mindom    maxdom   dfrange   modindx  label
0  0.275862  0.007812  0.007812  0.007812  0.000000  0.000000   male
1  0.250000  0.009014  0.007812  0.054688  0.046875  0.052632   male
2  0.271186  0.007990  0.007812  0.015625  0.007812  0.046512   male
3  0.250000  0.201497  0.007812  0.562500  0.554688  0.247119   male
4  0.266667  0.712812  0.007812  5.484375  5.476562  0.208274   male

[5 rows x 21 columns]
```

```
data.rename(columns = {'label':'Gender'}, inplace = True)
data
```

```
      meanfreq        sd    median       Q25       Q75       IQR
skew  \
0     0.059781  0.064241  0.032027  0.015071  0.090193  0.075122
12.863462
1     0.066009  0.067310  0.040229  0.019414  0.092666  0.073252
22.423285
2     0.077316  0.083829  0.036718  0.008701  0.131908  0.123207
30.757155
3     0.151228  0.072111  0.158011  0.096582  0.207955  0.111374
1.232831
4     0.135120  0.079146  0.124656  0.078720  0.206045  0.127325
1.101174
...        ...       ...       ...       ...       ...       ...
...
3163  0.131884  0.084734  0.153707  0.049285  0.201144  0.151859
1.762129
3164  0.116221  0.089221  0.076758  0.042718  0.204911  0.162193
0.693730
3165  0.142056  0.095798  0.183731  0.033424  0.224360  0.190936
1.876502
3166  0.143659  0.090628  0.184976  0.043508  0.219943  0.176435
1.591065
3167  0.165509  0.092884  0.183044  0.070072  0.250827  0.180756
1.705029

             kurt    sp.ent       sfm  ...  centroid   meanfun
minfun  \
0      274.402906  0.893369  0.491918  ...  0.059781  0.084279
0.015702
1      634.613855  0.892193  0.513724  ...  0.066009  0.107937
0.015826
2     1024.927705  0.846389  0.478905  ...  0.077316  0.098706
0.015656
3        4.177296  0.963322  0.727232  ...  0.151228  0.088965
0.017798
4        4.333713  0.971955  0.783568  ...  0.135120  0.106398
0.016931
...           ...       ...       ...  ...       ...       ...       .
```

```
..
3163     6.630383  0.962934  0.763182  ...  0.131884  0.182790
0.083770
3164     2.503954  0.960716  0.709570  ...  0.116221  0.188980
0.034409
3165     6.604509  0.946854  0.654196  ...  0.142056  0.209918
0.039506
3166     5.388298  0.950436  0.675470  ...  0.143659  0.172375
0.034483
3167     5.769115  0.938829  0.601529  ...  0.165509  0.185607
0.062257

        maxfun   meandom    mindom    maxdom   dfrange   modindx
Gender
0     0.275862  0.007812  0.007812  0.007812  0.000000  0.000000
male
1     0.250000  0.009014  0.007812  0.054688  0.046875  0.052632
male
2     0.271186  0.007990  0.007812  0.015625  0.007812  0.046512
male
3     0.250000  0.201497  0.007812  0.562500  0.554688  0.247119
male
4     0.266667  0.712812  0.007812  5.484375  5.476562  0.208274
male
...        ...       ...       ...       ...       ...       ...     .
..
3163  0.262295  0.832899  0.007812  4.210938  4.203125  0.161929
female
3164  0.275862  0.909856  0.039062  3.679688  3.640625  0.277897
female
3165  0.275862  0.494271  0.007812  2.937500  2.929688  0.194759
female
3166  0.250000  0.791360  0.007812  3.593750  3.585938  0.311002
female
3167  0.271186  0.227022  0.007812  0.554688  0.546875  0.350000
female

[3168 rows x 21 columns]
```

```
le = LabelEncoder()
data["Gender"]= le.fit_transform(data["Gender"])
data
```

```
      meanfreq        sd    median       Q25       Q75       IQR
skew  \
0     0.059781  0.064241  0.032027  0.015071  0.090193  0.075122
12.863462
1     0.066009  0.067310  0.040229  0.019414  0.092666  0.073252
22.423285
2     0.077316  0.083829  0.036718  0.008701  0.131908  0.123207
```

```
30.757155
3     0.151228  0.072111  0.158011  0.096582  0.207955  0.111374
1.232831
4     0.135120  0.079146  0.124656  0.078720  0.206045  0.127325
1.101174
...        ...       ...       ...       ...       ...       ...
...
3163  0.131884  0.084734  0.153707  0.049285  0.201144  0.151859
1.762129
3164  0.116221  0.089221  0.076758  0.042718  0.204911  0.162193
0.693730
3165  0.142056  0.095798  0.183731  0.033424  0.224360  0.190936
1.876502
3166  0.143659  0.090628  0.184976  0.043508  0.219943  0.176435
1.591065
3167  0.165509  0.092884  0.183044  0.070072  0.250827  0.180756
1.705029

             kurt    sp.ent       sfm  ...  centroid   meanfun
minfun  \
0      274.402906  0.893369  0.491918  ...  0.059781  0.084279
0.015702
1      634.613855  0.892193  0.513724  ...  0.066009  0.107937
0.015826
2     1024.927705  0.846389  0.478905  ...  0.077316  0.098706
0.015656
3        4.177296  0.963322  0.727232  ...  0.151228  0.088965
0.017798
4        4.333713  0.971955  0.783568  ...  0.135120  0.106398
0.016931
...           ...       ...       ...  ...       ...       ...       .
..
3163     6.630383  0.962934  0.763182  ...  0.131884  0.182790
0.083770
3164     2.503954  0.960716  0.709570  ...  0.116221  0.188980
0.034409
3165     6.604509  0.946854  0.654196  ...  0.142056  0.209918
0.039506
3166     5.388298  0.950436  0.675470  ...  0.143659  0.172375
0.034483
3167     5.769115  0.938829  0.601529  ...  0.165509  0.185607
0.062257

        maxfun   meandom    mindom    maxdom   dfrange   modindx
Gender
0     0.275862  0.007812  0.007812  0.007812  0.000000  0.000000
1
1     0.250000  0.009014  0.007812  0.054688  0.046875  0.052632
1
2     0.271186  0.007990  0.007812  0.015625  0.007812  0.046512
```

```
1
3     0.250000  0.201497  0.007812  0.562500  0.554688  0.247119
1
4     0.266667  0.712812  0.007812  5.484375  5.476562  0.208274
1
...        ...       ...       ...       ...       ...       ...     .
..
3163  0.262295  0.832899  0.007812  4.210938  4.203125  0.161929
0
3164  0.275862  0.909856  0.039062  3.679688  3.640625  0.277897
0
3165  0.275862  0.494271  0.007812  2.937500  2.929688  0.194759
0
3166  0.250000  0.791360  0.007812  3.593750  3.585938  0.311002
0
3167  0.271186  0.227022  0.007812  0.554688  0.546875  0.350000
0

[3168 rows x 21 columns]
```

```
X = data.drop(columns="Gender")
Y= data["Gender"]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.3, random_state = 1)

rfc = RandomForestClassifier(n_jobs=-1,max_features=
'sqrt' ,n_estimators=50, oob_score = True)

param_grid = {
    'n_estimators': [50, 300],
    #'max_features': ['auto', 'sqrt', 'log2'],
    #'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy']
}

CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
CV_rfc.fit(X, Y)

GridSearchCV(cv=5,
             estimator=RandomForestClassifier(max_features='sqrt',
                                              n_estimators=50,
n_jobs=-1,
                                              oob_score=True),
             param_grid={'criterion': ['gini', 'entropy'],
                         'n_estimators': [50, 300]})

print("Best Parameter and Estimator are : ")
print (CV_rfc.best_params_)

Best Parameter and Estimator are :
{'criterion': 'gini', 'n_estimators': 50}
```

AS6PR4

```
from sklearn.ensemble import AdaBoostClassifier
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import  classification_report
from sklearn.model_selection import GridSearchCV
```

```
data=pd.read_csv("seeds.csv")
data.head()
```

	Area	Perimeter	Compactness	Kernel.Length	Kernel.Width	\
0	15.26	14.84	0.8710	5.763	3.312	
1	14.88	14.57	0.8811	5.554	3.333	
2	14.29	14.09	0.9050	5.291	3.337	
3	13.84	13.94	0.8955	5.324	3.379	
4	16.14	14.99	0.9034	5.658	3.562	

	Asymmetry.Coeff	Kernel.Groove	Type
0	2.221	5.220	1
1	1.018	4.956	1
2	2.699	4.825	1
3	2.259	4.805	1
4	1.355	5.175	1

```
X = data.drop(columns="Type")
y=data["Type"]
X_train,X_test,y_train,y_test=
train_test_split(X,y,test_size=0.3,random_state=10)
```

```
dt=DecisionTreeClassifier(criterion='entropy',splitter='best',max_dept
h=1)
dt.fit(X_train,y_train)
y_pred = dt.predict(X_test)
DT_acc = accuracy_score(y_test,y_pred)
```

```
gaussian_nb=GaussianNB()
gaussian_nb.fit(X_train,y_train)
y_pred = gaussian_nb.predict(X_test)
bayes_acc = accuracy_score(y_test,y_pred)


print("Accuracy of Decision Tree Classifier is : ", DT_acc*100)
print("Accuracy of GaussianNB is : ", bayes_acc*100)


Accuracy of Decision Tree Classifier is :  60.0
Accuracy of GaussianNB is :  91.66666666666666

ada=AdaBoostClassifier(n_estimators=100,learning_rate=0.6)
ada.fit(X_train,y_train)
ada_pred=ada.predict(X_test)
acc_ada=round(accuracy_score(y_test,ada_pred),3)
print(acc_ada)


0.85

ada_gnb=AdaBoostClassifier(base_estimator=GaussianNB(),n_estimators=10
0,learning_rate=0.5)
ada_gnb.fit(X_train,y_train)
agnb_pred=ada_gnb.predict(X_test)
acc_agnb=round(accuracy_score(y_test,agnb_pred),3)
print(acc_agnb*100)


91.7

print ("Unique values are : ", data.Type.unique())

Unique values are :  [1 2 3]

print("Count of Unique Values are : ", data['Type'].value_counts())

Count of Unique Values are :  2    68
1    66
3    65
Name: Type, dtype: int64

print ("Percentage of Unique values are",
data['Type'].value_counts(normalize=True)*100)

Percentage of Unique values are 2    34.170854
1    33.165829
3    32.663317
Name: Type, dtype: float64

plt.figure(figsize=(10,2))
plt.barh(np.arange(4),[DT_acc,bayes_acc,acc_ada,acc_agnb],\
```

```
        tick_label=['Decision
Tree','GaussianNB','AdaBoost','AdaBoost(GaussianNB)'])
```

```
<BarContainer object of 4 artists>
```