In [1]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.ensemble import AdaBoostClassifier


pd.set_option("display.max_rows", None, "display.max_columns", None)

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [2]:

```
data = pd.read_csv("breast-cancer-data.csv")
```

In [3]:

```
data.describe()
```

Out[3]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.0000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.0963
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.0140
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.0526
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.0863
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.0958
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.1053
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.1634

In [11]:

```
data.head(10)
```

Out[11]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
0	1	17.99	10.38	122.80	1001.0	0.11840	
1	1	20.57	17.77	132.90	1326.0	0.08474	
2	1	19.69	21.25	130.00	1203.0	0.10960	
3	1	11.42	20.38	77.58	386.1	0.14250	
4	1	20.29	14.34	135.10	1297.0	0.10030	
5	1	12.45	15.70	82.57	477.1	0.12780	
6	1	18.25	19.98	119.60	1040.0	0.09463	
7	1	13.71	20.83	90.20	577.9	0.11890	
8	1	13.00	21.82	87.50	519.8	0.12730	
9	1	12.46	24.04	83.97	475.9	0.11860	

In [5]:

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

In [6]:

```
data.head(2)
```

Out[6]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
0	M	17.99	10.38	122.8	1001.0	0.11840	
1	M	20.57	17.77	132.9	1326.0	0.08474	

In [8]:

```
#data['diagnosis']
```

In [ ]:

```
#y = pd.get_dummies(data['diagnosis'])
```

In [ ]:

```
#y
```

In [12]:

```
data.diagnosis[data.diagnosis=='M'] = 1  #Cancerous
data.diagnosis[data.diagnosis=='B'] = 0
```

In [13]:

```
data.head(2)
```

Out[13]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
0	1	17.99	10.38	122.8	1001.0	0.11840	
1	1	20.57	17.77	132.9	1326.0	0.08474	

In [26]:

```
target = data['diagnosis'].astype('int8')
#features = data.iloc[:,1:]
```

In [22]:

```
features.head(2)
```

Out[22]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_me
0	17.99	10.38	122.8	1001.0	0.11840	0.277
1	20.57	17.77	132.9	1326.0	0.08474	0.078

In [23]:

```
target.head(2)
```

Out[23]:

```
0    1
1    1
Name: diagnosis, dtype: int8
```

In [24]:

```
xtrain,xtest,ytrain,ytest = train_test_split(features,
                                             target,
                                            test_size=0.25,
                                            random_state=40)
```

In [25]:

```
len(xtest)
```

Out[25]:

```
143
```

In [ ]:

```
#ytest
```

In [33]:

```
MinMax = MinMaxScaler()
```

In [34]:

```
scaled = StandardScaler()
```

In [35]:

```
#xtrain_minmax
```

Type *Markdown* and LaTeX: α^2

In [36]:

```
#xtrain_scaled
```

In [37]:

```
xtrain_minmax = MinMax.fit_transform(xtrain)
xtest_minmax = MinMax.transform(xtest)
```

In [38]:

```
xtrain_scaled = scaled.fit_transform(xtrain)
xtest_scaled = scaled.transform(xtest)
```

In [39]:

```
print(len(xtrain), len(ytrain), len(xtest), len(ytest) )
```

```
426 426 143 143
```

In [40]:

```
#print(type(xtrain), type(ytrain), type(xtest), type(ytest) )
```

In [48]:

```
clf = RandomForestClassifier(n_estimators=5,max_depth=10,random_state=0)
```

In [49]:

```
clf.fit(xtrain_scaled,ytrain)
```

Out[49]:

```
RandomForestClassifier(max_depth=10, n_estimators=5, random_state=0)
```

In [51]:

```
clf.score(xtrain_scaled,ytrain)
clf.score(xtest_scaled,ytest)
```

Out[51]:

```
0.9906103286384976
```

Out[51]:

```
0.9230769230769231
```

In [52]:

```
clf_pred = clf.predict(xtest_scaled)
```

In [56]:

```
#clf_pred
```

In [55]:

```
#ytest
```

In [ ]:

```
100 ... 96 , 100 ... 93 %

18th may ... 100
93%


100 ...96 %
```

In [ ]:

```
100 ... 93% ...

80 ... 96%
```

In [57]:

```
print(classification_report(ytest,clf_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.93      0.94        98
           1       0.85      0.91      0.88        45

    accuracy                           0.92       143
   macro avg       0.91      0.92      0.91       143
weighted avg       0.93      0.92      0.92       143
```

In [58]:

```
print(confusion_matrix(ytest,clf_pred))

#True positive   false Positive ..... 98
#false Negative  true Negative .....45


#actual result
98 ..... normal
45 .....  abnormal

#predicted result
TP - 91   , FP - 7 .....type 1 error

FN - 4 ...type 2 error    , TN - 41
```

```
[[91  7]
 [ 4 41]]
```

In [59]:

```
log = LogisticRegression()
```

In [60]:

```
log.fit(xtrain_scaled,ytrain)
```

Out[60]:

```
LogisticRegression()
```

In [61]:

```
log.score(xtest_scaled,ytest)
```

Out[61]:

```
0.972027972027972
```

In [62]:

```
log_pred = log.predict(xtest)
```

In [63]:

```
print(confusion_matrix(ytest,log_pred))
```

```
[[ 0 98]
 [ 0 45]]
```

In [64]:

```
print(classification_report(ytest,log_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        98
           1       0.31      1.00      0.48        45

    accuracy                           0.31       143
   macro avg       0.16      0.50      0.24       143
weighted avg       0.10      0.31      0.15       143
```

In [ ]:

```
1,2,3,4,5,6,7,8,9,10

143 ....
100 ....20...
80 ...
98 ...NORMAL ..............
45 ...ABNORMAL
......
```

In [ ]:

```
10 .....8 training and 2 for testing
```

In [ ]:

```
chennai ...


100 ... 80 ...20
```

In [ ]:

```
#### CROSS VALIDATION
```

In [ ]:

```
1,2,3,4,5,6,7,8,9,10
```

In [65]:

```
from sklearn.model_selection import cross_val_score
```

In [77]:

```
print(cross_val_score(clf,features,target,cv=10,scoring='accuracy'))
```

```
[0.96491228 0.87719298 0.89473684 0.96491228 0.96491228 0.98245614
 0.94736842 1.         0.94736842 0.96428571]
```

In [ ]:

```

```

In [ ]:

```
############## APPLYING PCA
```

In [ ]:

```
xtrain_scaled.shape
```

In [ ]:

```
xtrain.head(2)
```

In [ ]:

```
xtrain.shape
```

In [ ]:

```
from sklearn.decomposition import PCA
```

In [ ]:

```
pca = PCA(n_components=15)
```

In [ ]:

```
pca.fit(xtrain_scaled)
```

In [ ]:

```
x_pca = pca.transform(xtrain_scaled)
```

In [ ]:

```
x_pca.shape
```

In [ ]:

```
x_pca.shape
```

In [ ]:

```
x_pca
```

In [ ]:

```
### LABEL ENCODER
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
```

In [ ]:

```
#label = LabelEncoder()

#x =label.fit_transform(df['SaleCondition'])
```

In [ ]:

```

```

In [ ]:

```
data = pd.get_dummies()
```

In [ ]:

```

```

In [ ]:

```

```