Breast _cancer_classification and handling imbalance dataset

In [1]:

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
import warnings
warnings.filterwarnings("ignore")
```

Import Data

In [2]:

```
cancer_data=load_breast_cancer()
cancer_data
```

Out[2]:

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-0
1,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': 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,
```

In [ ]:

In [3]:

```
cancer_data_df=pd.DataFrame(data=cancer_data.data,columns=cancer_data.feature_names)
cancer_data_df['target']=cancer_data.target
cancer_data_df
```

Out[3]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	me symme
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.24
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.18
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.20
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.25
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.18
...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.17
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.17
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.15
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.23
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.15

569 rows × 31 columns

3.Data Understanding

In [4]:

```
cancer_data_df.shape
```

Out[4]:

```
(569, 31)
```

In [5]:

```
cancer_data_df.isna().sum()
```

Out[5]:

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

3.Data Understanding

In [6]:

```
cancer_data_df.shape
```

Out[6]:

```
(569, 31)
```

In [7]:

```
cancer_data_df.isna().sum()
```

Out[7]:

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

In [8]:

```
cancer_data_df.dtypes
```

Out[8]:

```
mean radius                float64
mean texture               float64
mean perimeter             float64
mean area                  float64
mean smoothness            float64
mean compactness           float64
mean concavity             float64
mean concave points        float64
mean symmetry              float64
mean fractal dimension     float64
radius error               float64
texture error              float64
perimeter error            float64
area error                 float64
smoothness error           float64
compactness error          float64
concavity error            float64
concave points error       float64
symmetry error             float64
fractal dimension error    float64
worst radius               float64
worst texture              float64
worst perimeter            float64
worst area                 float64
worst smoothness           float64
worst compactness          float64
worst concavity            float64
worst concave points       float64
worst symmetry             float64
worst fractal dimension    float64
target                       int32
dtype: object
```

4. Model Building

In [9]:

```
x=cancer_data_df.drop(labels='target',axis=1)
y=cancer_data_df[['target']]
```

In [29]:

```
om sklearn.model_selection import train_test_split
train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=12,stratify=y)
```

In [30]:

```
x_train.shape,y_train.shape
```

Out[30]:

```
((455, 30), (455, 1))
```

In [31]:

```
x_test.shape,y_test.shape
```

Out[31]:

```
((114, 30), (114, 1))
```

5.Model Training

In [32]:

```
from sklearn.ensemble import AdaBoostClassifier
adb_classifier=AdaBoostClassifier(base_estimator=None)
adb_classifier.fit(x_train,y_train)
```

Out[32]:

```
AdaBoostClassifier()
```

6.Model Testing || 7.Model Evaluation

Train data

In [33]:

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

In [34]:

```
y_pred_train=adb_classifier.predict(x_train)
print(confusion_matrix(y_train,y_pred_train))
print('Accuracy Score:',accuracy_score(y_train,y_pred_train))
```

```
[[170   0]
 [  0 285]]
Accuracy Score: 1.0
```

Test data

In [35]:

```
y_pred_test=adb_classifier.predict(x_test)
print(confusion_matrix(y_test,y_pred_test))
print('Accuracy Score:',accuracy_score(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
[[41  1]
 [ 1 71]]
Accuracy Score: 0.9824561403508771
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        42
           1       0.99      0.99      0.99        72

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114



===========================================================================
```

Handling imbalance dataset

In [36]:

```
cancer_data_df
```

Out[36]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	me symme
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1
...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1

569 rows × 31 columns

In [37]:

```
cancer_data_df['target'].value_counts()
```

Out[37]:

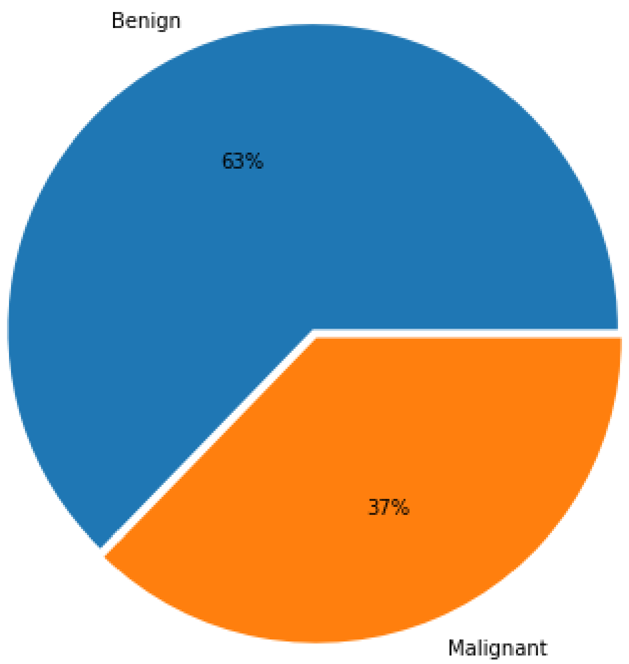
```
1    357
0    212
Name: target, dtype: int64
```

In [38]:

```
from matplotlib import pyplot as plt
plt.figure(figsize=(8,7))
plt.pie(x=cancer_data_df['target'].value_counts(),explode=[0.03,0],labels=['Benign','Malign
plt.show
```

Out[38]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

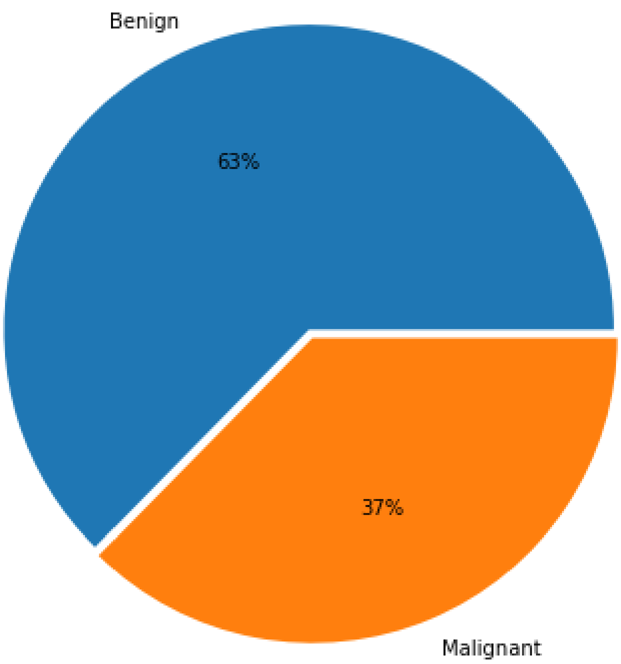


In [39]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=12,stratify=
```

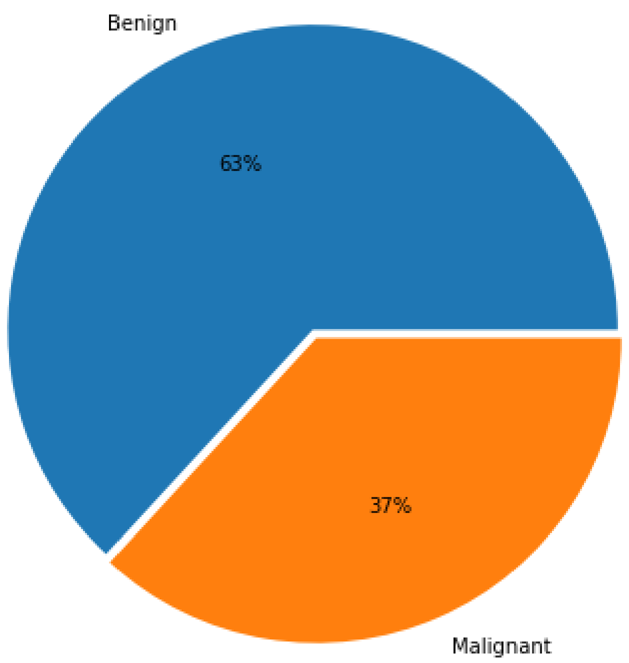
In [40]:

```
plt.figure(figsize=(8,7))
plt.pie(x=y_train.value_counts(),explode=[0.03,0],labels=['Benign','Malignant'],autopct='%1
plt.show()
```



In [41]:

```
plt.figure(figsize=(8,7))
plt.pie(x=y_test.value_counts(),explode=[0.03,0],labels=['Benign','Malignant'],autopct='%1.
plt.show()
```



In [42]:

```
from sklearn.linear_model import LogisticRegression
logistic_model=LogisticRegression(class_weight={0:3,1:1})
logistic_model.fit(x_train,y_train)
```

Out[42]:

```
LogisticRegression(class_weight={0: 3, 1: 1})
```

In [64]:

```
%%time
logistic_model.fit(x_train,y_train)
```

```
Wall time: 23.9 ms
```

Out[64]:

```
LogisticRegression(class_weight={0: 3, 1: 1})
```

In [43]:

```
y_pred=logistic_model.predict(x_train)
```

In [44]:

```
confusion_matrix(y_train,y_pred)
```

Out[44]:

```
array([[163,   7],
       [ 16, 269]], dtype=int64)
```

In [56]:

```
accuracy_score(y_train,y_pred)
```

Out[56]:

```
0.9494505494505494
```

In [57]:

```
from sklearn.tree import DecisionTreeClassifier
DecisionTreeClassifier()
```

Out[57]:

```
DecisionTreeClassifier()
```

==

Data preprocession for continous data

In [58]:

```
from sklearn.preprocessing import MinMaxScaler
minmax_scaler=MinMaxScaler()
minmax_scaler=minmax_scaler.fit_transform(x)
x_scaled=pd.DataFrame(data=minmax_scaler,columns=x.columns)
x_scaled
```

Out[58]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
0	0.521037	0.022658	0.545989	0.363733	0.593753	0.792037	0.703140	0.731113
1	0.643144	0.272574	0.615783	0.501591	0.289880	0.181768	0.203608	0.348757
2	0.601496	0.390260	0.595743	0.449417	0.514309	0.431017	0.462512	0.635686
3	0.210090	0.360839	0.233501	0.102906	0.811321	0.811361	0.565604	0.522863
4	0.629893	0.156578	0.630986	0.489290	0.430351	0.347893	0.463918	0.518390
...	...	...	...	...	...	...	...	...
564	0.690000	0.428813	0.678668	0.566490	0.526948	0.296055	0.571462	0.690358
565	0.622320	0.626987	0.604036	0.474019	0.407782	0.257714	0.337395	0.486630
566	0.455251	0.621238	0.445788	0.303118	0.288165	0.254340	0.216753	0.263519
567	0.644564	0.663510	0.665538	0.475716	0.588336	0.790197	0.823336	0.755467
568	0.036869	0.501522	0.028540	0.015907	0.000000	0.074351	0.000000	0.000000

569 rows × 30 columns

In [59]:

```
om sklearn.model_selection import train_test_split
train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=12,stratify=y)
```

In [60]:

```
from sklearn.linear_model import LogisticRegression
logistic_model=LogisticRegression(class_weight={0:3,1:1})
logistic_model.fit(x_train,y_train)
```

Out[60]:

```
LogisticRegression(class_weight={0: 3, 1: 1})
```

In [66]:

```
%%time
logistic_model.fit(x_train,y_train)
```

```
Wall time: 21 ms
```

Out[66]:

```
LogisticRegression(class_weight={0: 3, 1: 1})
```

In [67]:

```
y_pred=logistic_model.predict(x_train)
```

In [62]:

```
confusion_matrix(y_train,y_pred)
```

Out[62]:

```
array([[163,   7],
       [ 16, 269]], dtype=int64)
```

In [63]:

```
accuracy_score(y_train,y_pred)
```

Out[63]:

```
0.9494505494505494
```

In [ ]:

In [ ]:

In [ ]: