In [1]:
```
import numpy as np
import pandas as pd
```

In [13]:
```
df = pd.read_csv("/Users/dev/Personal/DS & AI Class Notes/Data Sets/Adaboos
```

col are 'Sex','Length','Whole weight','Viscera weight','Diameter','Height','Shucked weight','Shell weight','Rings'

In [34]:
```
df.columns = ['Sex','Length','Whole weight','Viscera weight','Diameter','He
```

In [40]:
```
df = df[['Length', 'Whole weight', 'Viscera weight', 'Diameter', 'Height',
       'Shucked weight', 'Shell weight', 'Rings','Sex']]
```

In [41]:
```
df
```

Out[41]:

	Length	Whole weight	Viscera weight	Diameter	Height	Shucked weight	Shell weight	Rings	Sex
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15	M
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7	M
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9	F
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10	M
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7	I
...	...	...	...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11	F
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10	M
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9	M
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10	F
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12	M

4177 rows × 9 columns

Checking Balance of DF

In [43]:
```
df["Sex"].value_counts()
```

Out[43]:
```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

In [44]:
```
def checkz(df):
    return df[df == 0].value_counts()
```

In [46]:
```
for i in df.columns:
    print(checkz(df[i]))
```

```
Series([], Name: Length, dtype: int64)
Series([], Name: Whole weight, dtype: int64)
0.0    2
Name: Viscera weight, dtype: int64
Series([], Name: Diameter, dtype: int64)
Series([], Name: Height, dtype: int64)
Series([], Name: Shucked weight, dtype: int64)
Series([], Name: Shell weight, dtype: int64)
Series([], Name: Rings, dtype: int64)
Series([], Name: Sex, dtype: int64)
```

In [47]:
```
df.columns
```

Out[47]:
```
Index(['Length', 'Whole weight', 'Viscera weight', 'Diameter', 'Height',
       'Shucked weight', 'Shell weight', 'Rings', 'Sex'],
      dtype='object')
```

Checking And Removing 0's

In [48]:
```
clist = ['Length', 'Whole weight', 'Viscera weight', 'Diameter', 'Height',
       'Shucked weight', 'Shell weight', 'Rings']
```

In [49]:
```
def zeroremove(df):
    m = round(df.mean(),2)
    df.replace(0,m,inplace = True)
```

In [50]:
```
for i in clist:
    zeroremove(df[i])
```

In [51]:
```
for i in df.columns:
    print(checkz(df[i]))
```

```
Series([], Name: Length, dtype: int64)
Series([], Name: Whole weight, dtype: int64)
Series([], Name: Viscera weight, dtype: int64)
Series([], Name: Diameter, dtype: int64)
Series([], Name: Height, dtype: int64)
Series([], Name: Shucked weight, dtype: int64)
Series([], Name: Shell weight, dtype: int64)
Series([], Name: Rings, dtype: int64)
Series([], Name: Sex, dtype: int64)
```

OD_Tech With The help of Skew

In [52]:
```
def odiqr(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    iqr = q3 - q1
    low = q1 - (1.5 * iqr)
    high = q3 + (1.5 * iqr)
    m = df.mean()
    df = df.apply(lambda x : m if x < low else (m if x > high else x ) )
    return df
```

In [53]:
```
def odmsd(df):
    m = round(df.mean(),2)
    s = round(df.std(),2)
    low = round(m-(3*s),2)
    high = round(m+(3*s),2)
    ft1 = df[df<low]
    ft2 = df[df>high]
    df = df.map(lambda x : low if x < low else (high if x > high else x ))
```

In [55]:
```
for i in clist:
    print(f'{i} is {df[i].skew()}')
```

```
Length is -0.639873268981801
Whole weight is -0.6091981423290918
Viscera weight is 3.1671068608877198
Diameter is 0.5309585632523087
Height is 0.7190979217612694
Shucked weight is 0.5918521514155083
Shell weight is 0.6209268251392077
Rings is 1.114101898355677
```

In [60]:
```
for i in clist:
    if df[i].skew() <= 0.5:
        odmsd(df[i])
    else:
        df[i] = odiqr(df[i])
```

In [61]:
```
for i in clist:
    print(f'{i} is {df[i].skew()}')
```

```
Length is -0.47226034516491344
Whole weight is -0.47042514918312894
Viscera weight is -0.1854164788342902
Diameter is 0.39235373317723926
Height is 0.42806642403527495
Shucked weight is 0.4557527876962757
Shell weight is 0.36801505496676334
Rings is 0.15139283233172263
```

Encoding of df["Sex"] with Label Encoder

In [63]:
```
from sklearn.preprocessing import LabelEncoder
```

In [64]:
```
le = LabelEncoder()
```

In [66]:
```
le.classes_
```

Out[66]:
```
array(['F', 'I', 'M'], dtype=object)
```

In [71]:
```
df = pd.concat([ df, pd.DataFrame(le.fit_transform(df["Sex"]),columns=["SE
```

In [72]:
```
df
```

Out[72]:

	Length	Whole weight	Viscera weight	Diameter	Height	Shucked weight	Shell weight	Rings	SEX
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15.0	2
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7.0	2
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9.0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10.0	2
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7.0	1
...	...	...	...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11.0	0
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10.0	2
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9.0	2
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10.0	0
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12.0	2

4177 rows × 9 columns

Splitting

In [73]:
```
X = df.drop("SEX",axis=1)
```

In [74]:
```
X.sample()
```

Out[74]:

	Length	Whole weight	Viscera weight	Diameter	Height	Shucked weight	Shell weight	Rings
2989	0.56	0.425	0.135	0.9415	0.509	0.2015	0.1975	9.0

In [75]:
```
y = df["SEX"]
```

In [76]:
```
y.sample()
```

Out[76]:
```
2791    2
Name: SEX, dtype: int64
```

In [77]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold , cross_val_score
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier , AdaBoostClassifier
```

In [78]:
```
Xtrain,Xtest,ytrain,ytest = train_test_split(X,y,test_size=.30)
```

In [79]:
```
X.shape , Xtrain.shape , Xtest.shape
```

Out[79]:
```
((4177, 8), (2923, 8), (1254, 8))
```

In [80]:
```
y.shape , ytrain.shape , ytest.shape
```

Out[80]:
```
((4177,), (2923,), (1254,))
```

In [81]:
```
kf = KFold(n_splits=11)
```

In [82]:
```
dct = DecisionTreeClassifier()
```

In [83]:
```
rfc = RandomForestClassifier()
```

In [84]:
```
algo = [ dct , rfc ]
```

Without Feature Scaling

In [85]:
```
for i in algo:
    i.fit(Xtrain,ytrain)
    s = i.score(Xtest,ytest)
    print(f'{i} = {s}')
```

```
DecisionTreeClassifier() = 0.48165869218500795
RandomForestClassifier() = 0.562200956937799
```

With Feature Scaling

In [86]:
```
ss = StandardScaler()
```

In [87]:
```
ss.fit(Xtrain)
```

Out[87]:
```
StandardScaler()
```

In [88]:
```
Xtrain_ss = ss.transform(Xtrain)
```

In [89]:
```
Xtest_ss = ss.transform(Xtest)
```

In [90]:
```
for i in algo:
    i.fit(Xtrain_ss,ytrain)
    s = i.score(Xtest_ss,ytest)
    print(f'{i} = {s}')
```

```
DecisionTreeClassifier() = 0.4904306220095694
RandomForestClassifier() = 0.5502392344497608
```

With Cross Validation

In [91]:
```
for i in algo:
    s = cross_val_score(i,X,y,cv = kf)
    print(f'{i} = {s.mean()}')
```

```
DecisionTreeClassifier() = 0.4876816351264344
RandomForestClassifier() = 0.524532577546048
```

Boosting

In [95]:
```
rfc1 = RandomForestClassifier(n_estimators=150,max_depth=2,max_leaf_nodes=3
```

In [96]:
```
abc = AdaBoostClassifier()
```

In [97]:
```
algo1 = [rfc1   , abc ]
```

In [98]:
```
for i in algo1:
    print(i)
```

```
RandomForestClassifier(max_depth=2, max_leaf_nodes=3, n_estimators=150)
AdaBoostClassifier()
```

Without Feature Scaling

In [99]:
```
for i in algo1:
    i.fit(Xtrain,ytrain)
    s = i.score(Xtest,ytest)
    print(f'{i} = {s}')
```

```
RandomForestClassifier(max_depth=2, max_leaf_nodes=3, n_estimators=150) = 0
.5223285486443381
AdaBoostClassifier() = 0.5350877192982456
```

With Feature Scaling

In [100…
```
for i in algo1:
    i.fit(Xtrain_ss,ytrain)
    s = i.score(Xtest_ss,ytest)
    print(f'{i} = {s}')
```

```
RandomForestClassifier(max_depth=2, max_leaf_nodes=3, n_estimators=150) = 0
.5215311004784688
AdaBoostClassifier() = 0.5350877192982456
```

With Cross Validation (Boosting)

In [101…
```
%%time
for i in algo1:
    s = cross_val_score(i,X,y,cv = kf)
    print(f'{i} = {s.mean()}')
```

```
RandomForestClassifier(max_depth=2, max_leaf_nodes=3, n_estimators=150) = 0
.5254970900506243
AdaBoostClassifier() = 0.533641160949868
CPU times: user 3.46 s, sys: 14.2 ms, total: 3.48 s
Wall time: 3.47 s
```

With GridSearch CV

In [102…
```
from sklearn.model_selection import GridSearchCV
```

In [108…
```
dic = { 'n_estimators' : [100,125,180],'criterion': ['gini', 'entropy'], 'r
       ,'min_samples_leaf' : [ 1,5]  }
```

In [109…
```
kf1 = KFold(n_splits=12)
```

In [110…
```
gvc = GridSearchCV(RandomForestClassifier(),param_grid=dic,cv = kf1)
```

In [111…
```
%%time
gvc.fit(X,y)
```

```
CPU times: user 2min 2s, sys: 754 ms, total: 2min 3s
Wall time: 2min 3s
```

Out[111…
```
GridSearchCV(cv=KFold(n_splits=12, random_state=None, shuffle=False),
             estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 10], 'min_samples_leaf': [1, 5
],
                         'n_estimators': [100, 125, 180]})
```

In [112…
```
gvc.best_params_
```

Out[112…
```
{'criterion': 'entropy',
 'max_depth': 10,
 'min_samples_leaf': 5,
 'n_estimators': 180}
```

In [113…
```
gvc.best_estimator_
```

Out[113…
```
RandomForestClassifier(criterion='entropy', max_depth=10, min_samples_leaf=
5,
                       n_estimators=180)
```

In [114…
```
gvc.best_score_
```

Out[114…
```
0.5540096773485271
```