In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```python
df = pd.read_csv('voice.csv')
df.head()
```

Out[3]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	0.000000	male
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	0.052632	male
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	0.046512	male
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	0.247119	male
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	0.208274	male

5 rows × 21 columns

In [4]:
```python
df.shape
```

Out[4]:
```
(3168, 21)
```

In [5]:
```python
df.isnull().sum()
```

Out[5]:
```
meanfreq    0
sd          0
median      0
Q25         0
Q75         0
IQR         0
skew        0
kurt        0
sp.ent      0
sfm         0
mode        0
centroid    0
meanfun     0
minfun      0
maxfun      0
meandom     0
mindom      0
maxdom      0
dfrange     0
modindx     0
label       0
dtype: int64
```

1) Since there is no null values so there is no replacement of any values.

In [7]:
```python
df.dtypes
```

Out[7]:
```
meanfreq    float64
sd          float64
median      float64
Q25         float64
Q75         float64
IQR         float64
skew        float64
kurt        float64
sp.ent      float64
sfm         float64
mode        float64
centroid    float64
meanfun     float64
minfun      float64
maxfun      float64
meandom     float64
mindom      float64
maxdom      float64
dfrange     float64
modindx     float64
label        object
dtype: object
```

Note

Here, we found that one of the data named 'label' has datatype of object so it is necessary to change its datatype

In [8]:
```python
print(df['label'].value_counts())
```

```
male      1584
female    1584
Name: label, dtype: int64
```

In [9]:
```python
from sklearn.preprocessing import LabelEncoder
```

In [10]:
```python
lb = LabelEncoder()
df['label'] = lb.fit_transform(df['label'])
```

In [11]:
```python
print(df['label'].value_counts())
```

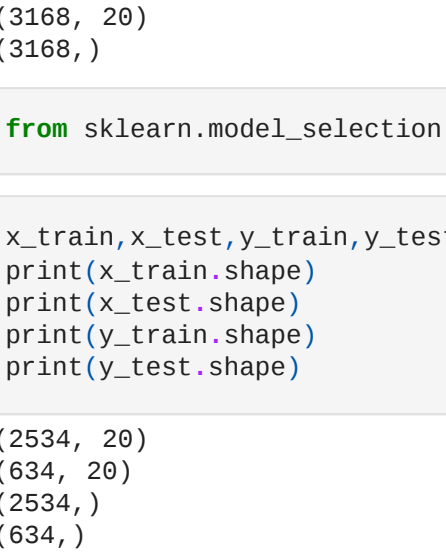
```
1    1584
0    1584
Name: label, dtype: int64
```

In [12]:
```python
df.dtypes
```

Out[12]:
```
meanfreq    float64
sd          float64
median      float64
Q25         float64
Q75         float64
IQR         float64
skew        float64
kurt        float64
sp.ent      float64
sfm         float64
mode        float64
centroid    float64
meanfun     float64
minfun      float64
maxfun      float64
meandom     float64
mindom      float64
maxdom      float64
dfrange     float64
modindx     float64
label         int32
dtype: object
```

2) Depicting the percentage distribution of 'label' on a piechart

In [15]:
```python
data =[1584, 1584]
label = ['Male','Female']
plt.pie(data, labels= label)
plt.show()
```



3) Considering all values as independent and 'label' as dependent value and splitting the data into training and test data with test size = 20%

In [18]:
```python
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
print(type(x))
print(type(y))
print(x.shape)
print(y.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
(3168, 20)
(3168,)
```

In [19]:
```python
from sklearn.model_selection import train_test_split
```

In [20]:
```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.20)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2534, 20)
(634, 20)
(2534,)
(634,)
```

In [22]:
```python
x_train.head()
```

Out[22]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx
2615	0.235271	0.029317	0.236563	0.223198	0.251933	0.028735	2.141051	7.623588	0.803982	0.124648	0.232554	0.235271	0.185272	0.048780	0.279070	1.992188	0.023438	10.242188	10.218750	0.151448
1547	0.165113	0.060920	0.133009	0.117278	0.227391	0.110112	2.806798	13.195741	0.910558	0.408569	0.121473	0.165113	0.114636	0.026882	0.270270	0.537109	0.004883	0.952148	0.947266	0.384255
804	0.173483	0.058914	0.183241	0.119797	0.227899	0.108101	1.280841	4.269679	0.926355	0.443394	0.121215	0.173483	0.095161	0.015717	0.163265	0.597470	0.007812	3.468750	3.460938	0.287246
2792	0.186230	0.032530	0.178776	0.170204	0.208571	0.038367	3.761459	22.198601	0.833996	0.214438	0.173061	0.186230	0.160490	0.050633	0.277457	1.940805	0.023438	9.937500	9.914062	0.164085
2351	0.234232	0.031842	0.238263	0.221650	0.252851	0.031201	2.062098	6.785469	0.826263	0.153974	0.246773	0.234232	0.175598	0.049587	0.279070	1.271365	0.023438	9.796875	9.773438	0.066294

In [23]:
```python
y_train.head()
```

Out[23]:
```
2615    0
1547    1
804     1
2792    0
2351    0
Name: label, dtype: int32
```

4) Different Classifier Models

a. Decsion Tree Classifier

In [24]:
```python
from sklearn.tree import DecisionTreeClassifier
```

In [97]:
```python
s1 = DecisionTreeClassifier(criterion = 'gini', max_depth = 18)
s1.fit(x_train,y_train)
```

Out[97]:
```
DecisionTreeClassifier(max_depth=18)
```

In [98]:
```python
print('Training Score', s1.score(x_train,y_train))
print('Testing Score', s1.score(x_test,y_test))
```

```
Training Score 1.0
Testing Score 0.9763406940063092
```

b. Random Forest Classifier

In [47]:
```python
from sklearn.ensemble import RandomForestClassifier
```

In [99]:
```python
s2 = RandomForestClassifier(n_estimators = 80, criterion = 'gini', max_depth = 18)
s2.fit(x_train,y_train)
```

Out[99]:
```
RandomForestClassifier(max_depth=18, n_estimators=80)
```

In [100…
```python
print('Training Score', s2.score(x_train,y_train))
print('Testing Score', s2.score(x_test,y_test))
```

```
Training Score 1.0
Testing Score 0.9858044164037855
```

c. KNN Classifier

In [101…
```python
from sklearn.neighbors import KNeighborsClassifier
```

In [102…
```python
s3 = KNeighborsClassifier(n_neighbors = 15)
s3.fit(x_train,y_train)
```

Out[102…
```
KNeighborsClassifier(n_neighbors=15)
```

In [103…
```python
print('Training Score', s3.score(x_train,y_train))
print('Testing Score', s3.score(x_test,y_test))
```

```
Training Score 0.7415153906866614
Testing Score 0.7113564668769716
```

d. Logistic Regression

In [104…
```python
from sklearn.linear_model import LogisticRegression
```

In [105…
```python
s4 = LogisticRegression(solver = 'liblinear')
s4.fit(x_train,y_train)
```

Out[105…
```
LogisticRegression(solver='liblinear')
```

In [106…
```python
print('Training Score', s4.score(x_train,y_train))
print('Testing Score', s4.score(x_test,y_test))
```

```
Training Score 0.9116022099447514
Testing Score 0.916403785488959
```

e. SVM Classifier

In [107…
```python
from sklearn.svm import SVC
```

In [108…
```python
s5 = SVC(kernel = 'linear', C=1)
s5.fit(x_train,y_train)
```

Out[108…
```
SVC(C=1, kernel='linear')
```

In [94]:
```python
print('Training Score', s5.score(x_train,y_train))
print('Testing Score', s5.score(x_test,y_test))
```

```
Training Score 0.9210734017363852
Testing Score 0.9227129337539433
```

5) Generating Confusion Matrix and Classification Report for all the Models

In [110…
```python
from sklearn.metrics import confusion_matrix,classification_report
```

In [121…
```python
def each_model(model, x_train,x_test,y_train,y_test):
    model.fit(x_train,y_train)
    ypredict = model.predict(x_test)
    conf_m = confusion_matrix(y_test,ypredict)
    print('Training Score',model.score(x_train,y_train))
    print('Testing Score',model.score(x_test,y_test))
    print('Predicted Values\n',ypredict)
    print('Confusion_Matrix\n',conf_m)
    print('Classification_Report\n',classification_report(y_test,ypredict))
```

a. Decision Tree Classifier

In [122…
```python
t1 = DecisionTreeClassifier(criterion='gini', max_depth= 18)
each_model(t1,x_train,x_test,y_train,y_test)
```

```
Training Score 1.0
Testing Score 0.9779179810725552
Predicted Values
 [0 0 1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 1
 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 0 0 0 1 1 0
 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 1 1 0 0 1 0
 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 0
 1 0 1 0 1 0 0 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 0
 0 0 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1
 0 0 1 1 0 0 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 0 0 1 1 0 1
 1 1 0 1 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 1 1 1 0 0 1 1 1 1 0 1 0 0 1
 1 0 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0
 1 1 1 1 0 1 0 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1
 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0
 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 1 1 0
 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 0 0
 0 0 1 0 0 1 1 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0
 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1
 0 1 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 1 1 1
 1 1 1 0 1]
Confusion_Matrix
 [[312   6]
 [  8 308]]
Classification_Report
               precision    recall  f1-score   support

           0       0.97      0.98      0.98       318
           1       0.98      0.97      0.98       316

    accuracy                           0.98       634
   macro avg       0.98      0.98      0.98       634
weighted avg       0.98      0.98      0.98       634
```

b. Random Forest Classifier

In [123…
```python
t2 = RandomForestClassifier(n_estimators=80,criterion='gini',max_depth=18)
each_model(t2,x_train,x_test,y_train,y_test)
```

```
Training Score 1.0
Testing Score 0.9842271293375394
Predicted Values
 [0 0 1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 1
 1 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 0 0 0 0 1 1 0
 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0 1 0
 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0
 1 0 1 0 1 0 0 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 0 0 1 0 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 0
 0 0 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 1 0 0 0 1 0 1 1 1 1 1 1 1
 0 0 1 1 0 0 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 0 0 1 1 0 1
 1 1 0 1 0 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 1 1 1 0 0 1 1 1 1 0 1 0 0 1
 1 0 0 0 1 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0
 1 1 1 1 0 1 0 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1
 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0
 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 1 0 0 1 0 1 1 1 1 0 0 0 1 1 0
 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 0 0
 0 0 1 0 0 1 1 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0
 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 0 1 1
 0 1 1 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1
 1 1 1 0 1]
Confusion_Matrix
 [[316   2]
 [  8 308]]
Classification_Report
               precision    recall  f1-score   support

           0       0.98      0.99      0.98       318
           1       0.99      0.97      0.98       316

    accuracy                           0.98       634
   macro avg       0.98      0.98      0.98       634
weighted avg       0.98      0.98      0.98       634
```

c. KNN Regression

In [127…
```python
t3 = KNeighborsClassifier(n_neighbors=15)
each_model(t3,x_train,x_test,y_train,y_test)
```

```
Training Score 0.7415153906866614
Testing Score 0.7113564668769716
Predicted Values
 [1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 1 1 0 1
 0 1 1 0 0 0 1 0 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0 0 1 1
 1 1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 0 1 1 1 0 1 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1
 0 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 1 1 0 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 0 0
 1 0 0 0 1 1 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1
 0 0 1 0 1 0 1 0 1 1 1 1 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0
 0 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 1 1 1 1 1
 0 1 0 1 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 0 1 1 0 0 1 0 1 0 0 0 1 1 0 0 0 0 1
 1 1 0 1 1 1 0 1 1 0 1 0 1 0 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 1
 1 0 1 1 1 1 1 1 0 0 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 1 0 0 1 1 0 1 1 0 1 0 0
 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0
 1 0 1 1 0 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 1
 0 1 0 1 0 0 0 0 0 1 0 1 1 1 1 1 0 1 1 1 0 0 1 0 0 1 1 1 1 1 0 0 0 0 0 1 0
 1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 1 0 1 0 0 1 1 1
 1 0 1 0 0 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0
 0 0 1 1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0 1 1 0 1 1 1
 0 1 1 1 1 0 1 1 1 0 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1
 1 0 1 0 1]
Confusion_Matrix
 [[205 113]
 [ 70 246]]
Classification_Report
               precision    recall  f1-score   support

           0       0.75      0.64      0.69       318
           1       0.69      0.78      0.73       316

    accuracy                           0.71       634
   macro avg       0.72      0.71      0.71       634
weighted avg       0.72      0.71      0.71       634
```

d. Logistic Regression

In [125…
```python
t4 = LogisticRegression(solver='liblinear')
each_model(t4,x_train,x_test,y_train,y_test)
```

```
Training Score 0.9116022099447514
Testing Score 0.916403785488959
Predicted Values
 [0 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 1
 1 1 0 0 0 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1 0 0 0 0 1 1 0
 1 0 1 0 0 0 0 1 1 0 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 0 1 1 1 0 0 1 1
 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0
 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1
 0 0 1 0 0 0 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 0
 0 0 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 1 0 0 0 1 0 1 1 1 1 1 1 1
 0 0 1 1 0 0 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 0 1 1 0 1
 1 1 0 1 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1
 1 0 0 0 1 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0
 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1
 1 0 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 0
 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 1 1 1 0 0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 0
 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 0 1 0 1 0 1
 1 0 1 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0
 1 0 1 1 0 1 0 1 1 0 1 0 1 0 0 0 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1
 0 1 1 1 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 0 0 0 1 1 1
 1 1 1 0 1]
Confusion_Matrix
 [[275  43]
 [ 10 306]]
Classification_Report
               precision    recall  f1-score   support

           0       0.96      0.86      0.91       318
           1       0.88      0.97      0.92       316

    accuracy                           0.92       634
   macro avg       0.92      0.92      0.92       634
weighted avg       0.92      0.92      0.92       634
```

e. SVM Classifier

In [129…
```python
t5 = SVC(kernel='linear',C= 1)
each_model(t5,x_train,x_test,y_train,y_test)
```

```
Training Score 0.9210734017363852
Testing Score 0.9227129337539433
Predicted Values
 [0 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 1
 1 1 0 0 0 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1 0 0 0 0 1 1 0
 1 0 1 0 1 0 0 1 1 0 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 0 1 1 1 0 0 1 1
 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0
 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1
 0 0 1 0 0 0 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 0
 0 0 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 1 1 0 0 0 1 0 1 1 1 1 1 1 1
 0 0 1 1 0 0 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 0 1 1 0 1
 1 1 0 1 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1
 1 0 0 0 1 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0
 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1
 1 0 0 1 0 0 0 1 1 0 0 0 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1 0
 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 0 1 1 0
 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 1 0 1
 1 0 1 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0
 1 0 1 1 0 1 0 1 1 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1
 0 1 1 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 0 0 0 1 1 1
 1 1 1 0 1]
Confusion_Matrix
 [[276  42]
 [  7 309]]
Classification_Report
               precision    recall  f1-score   support

           0       0.98      0.87      0.92       318
           1       0.88      0.98      0.93       316

    accuracy                           0.92       634
   macro avg       0.93      0.92      0.92       634
weighted avg       0.93      0.92      0.92       634
```

6) Best Accuracy Model

From the above 5 Models, Decision Tree Classifier and Random Forest Classifier has as an accuracy of 0.98. So either Decision Tree Classifier or Random Forest Classifier Model can be used to train the data