MINI PROJECT :

ABSTRACT:

The main focus of this project is to create a classification model to predict the gender (Male or Female) based on different acoustic parameters.

DATA SET :

In [97]:
```
import pandas as pd
data=pd.read_csv(r"C:\Users\zeush\Downloads\voice (1).csv")
data
```

Out[97]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	m
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
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
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
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
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

5 rows × 21 columns

3163	0.131884	0.084734	0.153707	0.049285	0.201144	0.151859	1.762129	6.630383	0.962934	0.763182	...	0.131884	0.182790	0.083770	0.262295	0.832899	0.007812	4.210938	4.203125	0
3164	0.116221	0.089221	0.076758	0.042718	0.204911	0.162193	0.693730	2.503954	0.960716	0.709570	...	0.116221	0.188980	0.034409	0.275862	0.909856	0.039062	3.679688	3.640625	0
3165	0.142056	0.095798	0.183731	0.033424	0.224360	0.190936	1.876502	6.604509	0.946854	0.654196	...	0.142056	0.209518	0.039596	0.275862	0.494271	0.007812	2.937500	2.929688	0
3166	0.143659	0.090628	0.184970	0.043508	0.219943	0.176435	1.591065	5.388298	0.950436	0.675470	...	0.143659	0.172335	0.034483	0.250000	0.791360	0.007812	3.593750	3.585938	0
3167	0.165509	0.092884	0.183044	0.070072	0.250827	0.180756	1.705029	5.769115	0.938829	0.601529	...	0.165509	0.185667	0.062257	0.271186	0.227022	0.007812	0.554688	0.546875	0

3168 rows × 21 columns

DATA CLEANING:

In [98]:
```
data.isnull().sum()
#AS WE CAN SEE THERE ARE NO NULL VALUES GIVEN IN THE DATA SET
```

Out[98]:
```
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
```

In [99]:
```
data.columns
```

Out[99]:
```
Index(['meanfreq', 'sd', 'median', 'Q25', 'Q75', 'IQR', 'skew', 'kurt',
       'sp.ent', 'sfm', 'mode', 'centroid', 'meanfun', 'minfun', 'maxfun',
       'meandom', 'mindom', 'maxdom', 'dfrange', 'modindx', 'label'],
      dtype='object')
```

In [100...
```
data.dtypes
# As we can see all the training variables are same datatype. So, we need not to use lable_encoder here.
```

Out[100...
```
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
```

In [101...
```
data.shape
```

Out[101...
```
(3168, 21)
```

In [102...
```
x=data.iloc[:,:20].values
y=data.iloc[:,20].values
x
```

Out[102...
```
array([[0.05978098, 0.06424127, 0.03202691, ..., 0.0078125 , 0.        ,
        0.        ],
       [0.06600874, 0.06731003, 0.04022873, ..., 0.0546875 , 0.046875  ,
        0.0526316 ],
       [0.07731155 , 0.08382942, 0.03671846, ..., 0.015625  , 0.0078125 ,
        0.04651163],
       ...,
       [0.14205626, 0.09579843, 0.18373124, ..., 2.9375    , 2.9296875 ,
        0.19475862],
       [0.14365874, 0.09062826, 0.18497617, ..., 3.59375   , 3.5859375 ,
        0.31100218],
       [0.16550895, 0.09288354, 0.18304392, ..., 0.5546875 , 0.546875  ,
        0.35      ]])
```

In [103...
```
y
```

Out[103...
```
array(['male', 'male', 'male', ..., 'female', 'female', 'female'],
      dtype=object)
```

DATA VISUALIZATION :

In [104...
```
import matplotlib.pyplot as plt
#plt.pie(data['label'].value_counts())
mylabels = ["Male","Female"]
plt.pie(data['label'].value_counts(), labels = mylabels, autopct='%1.1f%%', startangle = 90)
plt.show()
```

This is a pie chart representation of label in the given data set.

DATA PREPROCESSING :

In [105...
```
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)
```

In [106...
```
x_train.shape
```

Out[106...
```
(2534, 20)
```

In [107...
```
y_train.shape
```

Out[107...
```
(2534,)
```

In [108...
```
x_test.shape
```

Out[108...
```
(634, 20)
```

In [109...
```
y_test.shape
```

Out[109...
```
(634,)
```

MODELBUILDING :

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

Model 1 : (Decision Tree Classifier)

In [111...
```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(criterion='gini')
model.fit(x_train,y_train)
ypred=model.predict(x_test)
print(' confusion matrix of DT :',confusion_matrix(y_test,ypred))
print()
print('classification_report of DT :',classification_report(y_test,ypred))
print()
print('Accuracy score of DT :',accuracy_score(y_test,ypred)*100)
```

```
 confusion matrix of DT : [[323   9]
 [ 15 287]]

classification_report of DT :              precision    recall  f1-score   support

      female       0.96      0.97      0.96       332
        male       0.97      0.95      0.96       302

    accuracy                           0.96       634
   macro avg       0.96      0.96      0.96       634
weighted avg       0.96      0.96      0.96       634


Accuracy score of DT : 96.21451104100946
```

Model 2 : (Random Forest Classifier)

In [112...
```
from sklearn.ensemble import RandomForestClassifier
modelrf=RandomForestClassifier()
modelrf.fit(x_train,y_train)
ypredrf=modelrf.predict(x_test)
print(' confusion matrix of RF :',confusion_matrix(y_test,ypredrf))
print()
print('classification_report of RF :',classification_report(y_test,ypredrf))
print()
print('Accuracy score of RF :',accuracy_score(y_test,ypredrf)*100)
```

```
 confusion matrix of RF : [[325   7]
 [ 11 291]]

classification_report of RF :              precision    recall  f1-score   support

      female       0.97      0.98      0.97       332
        male       0.98      0.96      0.97       302

    accuracy                           0.97       634
   macro avg       0.97      0.97      0.97       634
weighted avg       0.97      0.97      0.97       634


Accuracy score of RF : 97.16088328075709
```

Model 3 : (KNN Classifier)

In [113...
```
from sklearn.neighbors import KNeighborsClassifier
modelknn=KNeighborsClassifier(n_neighbors=3)
modelknn.fit(x_train,y_train)
ypredknn=modelknn.predict(x_test)
print(' confusion matrix of KNN :',confusion_matrix(y_test,ypredknn))
print()
print('classification_report of KNN :',classification_report(y_test,ypredknn))
print()
print('Accuracy score of KNN :',accuracy_score(y_test,ypredknn)*100)
```

```
 confusion matrix of KNN : [[229 103]
 [ 80 222]]

classification_report of KNN :              precision    recall  f1-score   support

      female       0.74      0.69      0.71       332
        male       0.68      0.74      0.71       302

    accuracy                           0.71       634
   macro avg       0.71      0.71      0.71       634
weighted avg       0.71      0.71      0.71       634


Accuracy score of KNN : 71.13564668769716
```

Model 4 : (Logistic Regression)

In [114...
```
from sklearn.linear_model import LogisticRegression
modellr=LogisticRegression(solver='liblinear')
modellr.fit(x_train,y_train)
ypredlr=modellr.predict(x_test)
print(' confusion matrix of LR :',confusion_matrix(y_test,ypredlr))
print()
print('classification_report of LR :',classification_report(y_test,ypredlr))
print()
print('Accuracy score of LR :',accuracy_score(y_test,ypredlr)*100)
```

```
 confusion matrix of LR : [[277  55]
 [ 13 289]]

classification_report of LR :              precision    recall  f1-score   support

      female       0.96      0.83      0.89       332
        male       0.84      0.96      0.89       302

    accuracy                           0.89       634
   macro avg       0.90      0.90      0.89       634
weighted avg       0.90      0.89      0.89       634


Accuracy score of LR : 89.27444794952682
```

Model 5 : (SVM Classifier)

In [115...
```
from sklearn.svm import SVC
modelsvm=SVC()
modelsvm.fit(x_train,y_train)
ypredsvm=modelsvm.predict(x_test)
print(' confusion matrix of SVM :',confusion_matrix(y_test,ypredsvm))
print()
print('classification_report of SVM :',classification_report(y_test,ypredsvm))
print()
print('Accuracy score of SVM :',accuracy_score(y_test,ypredsvm)*100)
```

```
 confusion matrix of SVM : [[169 163]
 [ 54 248]]

classification_report of SVM :              precision    recall  f1-score   support

      female       0.76      0.51      0.61       332
        male       0.60      0.82      0.70       302

    accuracy                           0.66       634
   macro avg       0.68      0.67      0.65       634
weighted avg       0.68      0.66      0.65       634


Accuracy score of SVM : 65.77287066246056
```

Conclusion :

Random Forest is the best performing model in terms of accuracy