Voice Categorization - Case Study

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Set: Voice Data Set

This database was created to identify a voice as male or female, based upon acoustic properties of the voice and speech. The dataset consists of 3800 recorded voice samples. The voice samples are pre-processed by acoustic analysis in R using the seewave and tuneR packages.

The following acoustic properties of each voice are measured and included within the CSV:

- **meanfreq**: mean frequency (in kHz)
- **sd**: standard deviation of frequency
- **median**: median frequency (in kHz)
- **Q25**: first quantile (in kHz)
- **Q75**: third quantile (in kHz)
- **IQR**: interquantile range (in kHz)
- **skew**: skewness (see note in specprop description)
- **kurt**: kurtosis (see note in specprop description)
- **sp.ent**: spectral entropy
- **sfm**: spectral flatness
- **mode**: mode frequency
- **centroid**: frequency centroid (see specprop)
- **peakf**: peak frequency (frequency with highest energy)
- **meanfun**: average of fundamental frequency measured across acoustic signal
- **minfun**: minimum fundamental frequency measured across acoustic signal
- **maxfun**: maximum fundamental frequency measured across acoustic signal
- **meandom**: average of dominant frequency measured across acoustic signal
- **mindom**: minimum of dominant frequency measured across acoustic signal
- **maxdom**: maximum of dominant frequency measured across acoustic signal
- **dfrange**: range of dominant frequency measured across acoustic signal
- **modindx**: modulation index. Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range
- **label**: male or female

In [2]:
```python
df = pd.read_csv('voice-classification.csv')
df.head()
```

Out[2]:

	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 [3]:
```python
# Check the no. of records
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3168 entries, 0 to 3167
Data columns (total 21 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   meanfreq  3168 non-null   float64
 1   sd        3168 non-null   float64
 2   median    3168 non-null   float64
 3   Q25       3168 non-null   float64
 4   Q75       3168 non-null   float64
 5   IQR       3168 non-null   float64
 6   skew      3168 non-null   float64
 7   kurt      3168 non-null   float64
 8   sp.ent    3168 non-null   float64
 9   sfm       3168 non-null   float64
 10  mode      3168 non-null   float64
 11  centroid  3168 non-null   float64
 12  meanfun   3168 non-null   float64
 13  minfun    3168 non-null   float64
 14  maxfun    3168 non-null   float64
 15  meandom   3168 non-null   float64
 16  mindom    3168 non-null   float64
 17  maxdom    3168 non-null   float64
 18  dfrange   3168 non-null   float64
 19  modindx   3168 non-null   float64
 20  label     3168 non-null   object 
dtypes: float64(20), object(1)
memory usage: 519.9+ KB
```

In [4]:
```python
# Check the Basic Distribution of Data
df.describe()
```

Out[4]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfun	minfun	maxfun	meandom	mind
count	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000
mean	0.180907	0.057126	0.185621	0.140456	0.224765	0.084309	3.140168	36.568461	0.895127	0.408216	0.165282	0.180907	0.142807	0.036802	0.258842	0.829211	0.052
std	0.029918	0.016652	0.036360	0.048680	0.023639	0.042783	4.240529	134.928661	0.044980	0.177521	0.077203	0.029918	0.032304	0.019220	0.030077	0.525205	0.063
min	0.039363	0.018363	0.010975	0.000229	0.042946	0.014558	0.141735	2.068455	0.738651	0.036876	0.000000	0.039363	0.055565	0.009775	0.103093	0.007812	0.004
25%	0.163662	0.041954	0.169593	0.111087	0.208747	0.042560	1.649569	5.669547	0.861811	0.258041	0.118016	0.163662	0.116998	0.018223	0.253968	0.419828	0.007
50%	0.184838	0.059155	0.190032	0.140286	0.225684	0.094280	2.197101	8.318463	0.901767	0.396335	0.186599	0.184838	0.140519	0.046110	0.271186	0.765795	0.023
75%	0.199146	0.067020	0.210618	0.175939	0.243660	0.114175	2.931694	13.648905	0.928713	0.533676	0.221104	0.199146	0.169581	0.047904	0.277457	1.177166	0.070
max	0.251124	0.115273	0.261224	0.247347	0.273469	0.252225	34.725453	1309.612887	0.981997	0.842936	0.280000	0.251124	0.237636	0.204082	0.279114	2.957682	0.458

Checking whether there is any null values

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
```

Get Shape and Distribution

In [6]:
```python
print ( "Shape of Data:" , df.shape)
print("Total number of labels: {}".format(df.shape[0]))
print("Number of male: {}".format(df[df.label == 'male'].shape[0]))
print("Number of female: {}".format(df[df.label == 'female'].shape[0]))
```

```
Shape of Data: (3168, 21)
Total number of labels: 3168
Number of male: 1584
Number of female: 1584
```

In [7]:
```python
X=df.iloc[:, :-1]
print (df.shape)
print (X.shape)
```

```
(3168, 21)
(3168, 20)
```

Converting label column (male/female) to 1/0

In [8]:
```python
from sklearn.preprocessing import LabelEncoder
```

In [9]:
```python
# Get All rows, but only last column
y=df.iloc[:,-1]
# Encode label category
# male -> 1
# female -> 0

gender_encoder = LabelEncoder()
y = gender_encoder.fit_transform(y)
y
```

Out[9]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [10]:
```python
# Scale the data to be between -1 and 1
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
```

Train Test Split

Split your data into a training set and a testing set.

In [11]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=100)
```

Train a Model

Now, it's time to train a Support Vector Machine Classifier.

Call the SVC() model from sklearn and fit the model to the training data.

In [12]:
```python
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import classification_report,confusion_matrix
```

In [13]:
```python
# ALL Default hyperparameters
svc_model=SVC()
svc_model.fit(X_train,y_train)
y_pred=svc_model.predict(X_test)
```

In [14]:
```python
#Percentage of observations used to build the SVC (Support Vectors)
svc_model.support_vectors_.shape[0]/df.shape[0]*100
```

Out[14]:
```
9.785353535353536
```

Model Evaluation

In [15]:
```python
print('Accuracy Score:')
print(metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy Score:
0.9737118822292324
```

In [16]:
```python
print(confusion_matrix(y_test,y_pred))
```

```
[[458  13]
 [ 12 468]]
```

In [17]:
```python
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       471
           1       0.97      0.97      0.97       480

    accuracy                           0.97       951
   macro avg       0.97      0.97      0.97       951
weighted avg       0.97      0.97      0.97       951
```

Let's see if you can tune the parameters to get even better (Probably, you would be satisfied with these results in real life because the data set is quite small, but the idea is you should practice using GridSearch.)

Gridsearch

In [18]:
```python
from sklearn.model_selection import GridSearchCV
```

Create a dictionary called param_grid and fill out some parameters for C.

In [19]:
```python
param_grid = {'C': [0.1,1, 10, 100], 'kernel': ['poly', 'linear', 'rbf', 'sigmoid']}
```

In [20]:
```python
grid = GridSearchCV(SVC(),param_grid)
grid.fit(X_train,y_train)
```

Out[20]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100],
                         'kernel': ['poly', 'linear', 'rbf', 'sigmoid']})
```

Now, take that grid model and create some predictions using the test set and create classification reports and confusion matrices for them.

In [21]:
```python
grid.best_params_
```

Out[21]:
```
{'C': 10, 'kernel': 'rbf'}
```

In [22]:
```python
grid_predictions = grid.predict(X_test)
```

In [23]:
```python
print('Accuracy Score:')
print(metrics.accuracy_score(y_test,grid_predictions))
```

```
Accuracy Score:
0.9779179810725552
```

In [24]:
```python
print(confusion_matrix(y_test,grid_predictions))
```

```
[[460  11]
 [ 10 470]]
```

In [25]:
```python
print(classification_report(y_test,grid_predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       471
           1       0.98      0.98      0.98       480

    accuracy                           0.98       951
   macro avg       0.98      0.98      0.98       951
weighted avg       0.98      0.98      0.98       951
```