In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import seaborn as sns
from sklearn import metrics
```

In [2]:
```
# 1.We will use acoustic features to distinguish a male voice from female.
# Load the dataset from “voice.csv”,identify the target variable and do a one-hot encoding for the same.
# Split the dataset in train-test with 20% of the data kept aside for testing
df_voices = pd.read_csv("voice.csv")
df_voices.head()
```

Out[2]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.en
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.89336
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.89219
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.84638
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.96332
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.97195

5 rows × 21 columns

In [3]:
```
df_voices["label"] = df_voices["label"].map({'male':0, 'female':1})

X = df_voices.iloc[:,0:19]
Y = df_voices["label"]
```

In [4]:
```
# 2.Fit a logistic regression model and measure the accuracy on the test set
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, rand
om_state=10)

linear_model = LogisticRegression()
linear_model.fit(x_train, y_train)

predicted_data = linear_model.predict(x_test)

metrics.accuracy_score(predicted_data, y_test)
```

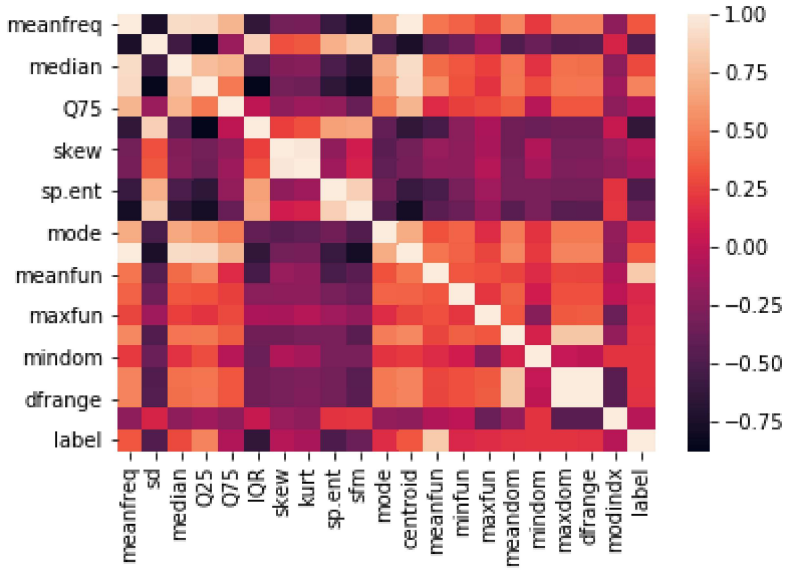
```
C:\Users\hp\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:94
0: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[4]:
```
0.9211356466876972
```

In [5]:
```
# 3.Compute the correlation matrix that describes the dependence between all p
redictors and identify the
# predictors that are highly correlated.  Plot the correlation matrix using se
aborn heatmap.
corr = df_voices.corr()
sns.heatmap(corr)
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x269e4720bc8>
```



In [14]:
```
df_voices.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   int64  
dtypes: float64(20), int64(1)
memory usage: 519.9 KB
```

In [18]:

```
# 4. Based on correlation remove those predictors that are correlated and fit
 a logistic regression model again
# and compare the accuracy with that of previous model
X = X.drop("median", axis=1)
X = X.drop("Q25", axis=1)
X = X.drop("centroid", axis=1)
X = X.drop("dfrange", axis=1)

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.20, ra
ndom_state=10)

linear_model = LogisticRegression()
linear_model.fit(x_train, y_train)

predicted_data = linear_model.predict(x_test)

metrics.accuracy_score(predicted_data)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-18-d7754f2510a5> in <module>
      1 # 4. Based on correlation remove those predictors that are correlated
and fit a logistic regression model again
      2 # and compare the accuracy with that of previous model
----> 3 X = X.drop("median", axis=1)
      4 X = X.drop("Q25", axis=1)
      5 X = X.drop("centroid", axis=1)

~\anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axi
s, index, columns, level, inplace, errors)
   3995             level=level,
   3996             inplace=inplace,
-> 3997             errors=errors,
   3998         )
   3999

~\anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, ax
is, index, columns, level, inplace, errors)
   3934         for axis, labels in axes.items():
   3935             if labels is not None:
-> 3936                 obj = obj._drop_axis(labels, axis, level=level, error
s=errors)
   3937
   3938         if inplace:

~\anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labe
ls, axis, level, errors)
   3968                 new_axis = axis.drop(labels, level=level, errors=erro
rs)
   3969             else:
-> 3970                 new_axis = axis.drop(labels, errors=errors)
   3971             result = self.reindex(**{axis_name: new_axis})
   3972

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, label
s, errors)
   5016         if mask.any():
   5017             if errors != "ignore":
-> 5018                 raise KeyError(f"{labels[mask]} not found in axis")
   5019             indexer = indexer[~mask]
   5020         return self.delete(indexer)

KeyError: "['median'] not found in axis"
```

In [ ]: