In [1]:

```python
import pandas as pd
VoiceData=pd.read_csv('voice.csv')#reading the data
```

In [2]:

```python
VoiceData.head()
```

Out[2]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568

5 rows × 21 columns

In [3]:

```python
VoiceData.isnull().sum()#checking for null values
#Eeven though the null values are clearly visisble in the dataset the isnull function
```

Out[3]:

```
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 [4]:

```python
X=VoiceData.iloc[:,:-1]#independent features
y=VoiceData.iloc[:,-1]#dependent features
```

In [5]:

```python
X.head()
print(X.shape)
```

```
(3168, 20)
```

In [6]:

```python
print(y.shape)
```

```
(3168,)
```

In [7]:

```python
y.value_counts()
```

Out[7]:

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

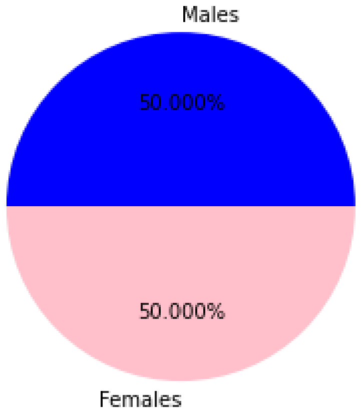
In [8]:

```python
import matplotlib.pyplot as plt
countMale=1584
countFemale=1584
values=[countMale,countFemale]
names=['Males','Females']
```

```
clr=['blue','pink']
plt.pie(values,labels=names,autopct='%2.3f%%',colors=clr)
```

Out[8]:

```
([<matplotlib.patches.Wedge at 0x1724c69e2c0>,
  <matplotlib.patches.Wedge at 0x1724c69ea10>],
 [Text(6.735557395310444e-17, 1.1, 'Males'),
  Text(-2.0206672185931328e-16, -1.1, 'Females')],
 [Text(3.6739403974420595e-17, 0.6, '50.000%'),
  Text(-1.1021821192326178e-16, -0.6, '50.000%')])
```



In [9]:

```
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)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

Logistic Regression Model:

In [10]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,classification_report
m1=LogisticRegression(solver='lbfgs', max_iter=500)
m1.fit(X_train,y_train)
Y_pred_m1=m1.predict(X_test)
cm=confusion_matrix(y_test,Y_pred_m1)
print('The Confusion Matrix is:')
print(cm)
print('\n')
print('The Classification report:')
print(classification_report(y_test,Y_pred_m1))
```

```
The Confusion Matrix is:
[[272  41]
 [ 10 311]]


The Classification report:
              precision    recall  f1-score   support

      female       0.96      0.87      0.91       313
        male       0.88      0.97      0.92       321

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

KNN Classifier Model:

In [11]:

```python
from sklearn.neighbors import KNeighborsClassifier
m2=KNeighborsClassifier(n_neighbors=3)
m2.fit(X_train,y_train)
Y_pred_m2=m2.predict(X_test)
cm=confusion_matrix(y_test,Y_pred_m2)
print('The Confusion Matrix is:')
print(cm)
print('\n')
print('The Classification report:')
print(classification_report(y_test,Y_pred_m2))
```

```
The Confusion Matrix is:
[[207 106]
 [ 86 235]]


The Classification report:
              precision    recall  f1-score   support

      female       0.71      0.66      0.68       313
        male       0.69      0.73      0.71       321

    accuracy                           0.70       634
   macro avg       0.70      0.70      0.70       634
weighted avg       0.70      0.70      0.70       634
```

Decision Tree Classifier:

In [12]:

```python
from sklearn.tree import DecisionTreeClassifier
m3=DecisionTreeClassifier(criterion='gini',max_depth=1)
m3.fit(X_test,y_test)
Y_pred_m3=m3.predict(X_test)
cm=confusion_matrix(y_test,Y_pred_m3)
print('The Confusion Matrix is:')
print(cm)
print('\n')
print('The Classification report:')
print(classification_report(y_test,Y_pred_m3))
```

```
The Confusion Matrix is:
[[294  19]
 [ 12 309]]


The Classification report:
              precision    recall  f1-score   support

      female       0.96      0.94      0.95       313
        male       0.94      0.96      0.95       321

    accuracy                           0.95       634
   macro avg       0.95      0.95      0.95       634
weighted avg       0.95      0.95      0.95       634
```

Random Forest Classifier

In [13]:

```python
from sklearn.ensemble import RandomForestClassifier
m4=RandomForestClassifier(n_estimators=55,criterion='gini',max_depth=5)
m4.fit(X_test,y_test)
```

```
Y_pred_m4=m4.predict(X_test)
cm=confusion_matrix(y_test,Y_pred_m4)
print('The Confusion Matrix is:')
print(cm)
print('\n')
print('The Classification report:')
print(classification_report(y_test,Y_pred_m4))
```

```
The Confusion Matrix is:
[[308   5]
 [  1 320]]


The Classification report:
              precision    recall  f1-score   support

      female       1.00      0.98      0.99       313
        male       0.98      1.00      0.99       321

    accuracy                           0.99       634
   macro avg       0.99      0.99      0.99       634
weighted avg       0.99      0.99      0.99       634
```

Support Vector Machine(SVM)

In [14]:

```
from sklearn.svm import SVC
m5=SVC(kernel='linear',C=1)
m5.fit(X_train,y_train)
Y_pred_m5=m5.predict(X_test)
cm=confusion_matrix(y_test,Y_pred_m5)
print('The Confusion Matrix is:')
print(cm)
print('\n')
print('The Classification report:')
print(classification_report(y_test,Y_pred_m5))
```

```
The Confusion Matrix is:
[[277  36]
 [  6 315]]


The Classification report:
              precision    recall  f1-score   support

      female       0.98      0.88      0.93       313
        male       0.90      0.98      0.94       321

    accuracy                           0.93       634
   macro avg       0.94      0.93      0.93       634
weighted avg       0.94      0.93      0.93       634
```