PROJECT STATUS

Problem Statement:

Gender Recognition system from audio files using FFT with Artificial Neural Networks.

Status:

4. Dataset Preprocessing

Modules Used:

1. Numpy
2. Pandas
3. scikit-learn
4. Matplotlib

4.1 Dealing with NAN value:

This helps to deal with the missing values which is presents in the dataframe.To deal with that , it is best to use the mean of the columns where the the data is missing .

Implementation:

```
#dealing with missing values
@classmethod
def nanval(self):
    if (self.df.isnull().values.any()) == True:
        null_columns=train.columns[train.isnull().any()]
        for i in null_columns:
            self.df[i] = self.df[i].fillna(self.df[i].mean())
    return self.df
```

4.2 Label Encoding:

As we know that all the algorithm computes the numeric values , but the dataset contans the textual data so it will be difficult to deal with those.for those kind of cases we are using encoding to map words to a numeric vectors .

For Eg.

“Male” -----: 1
“Female” --: 0

Implementation:

```
#encoding the dataset
@classmethod
def encoding(self):
    self.x = self.df.iloc[:,:-1].values
    self.y = self.df.iloc[:,-1].values
    le = preprocessing.LabelEncoder()
    self.y = le.fit_transform(self.y)

    return self.x, self.y
```

4.3 Scaling Features

The Large values in computation requires a lots of time ,in these cases feature scaling plays a vital role which helps the data ot be in the scaled form which helps the computation to be more faster and efficient.

Types:

1.MinMax Scaler
2.Standard Scaler

Implementation:

In this project i will be using MinMax Scaler, The formula of minmax scaler are as follows:

$$z_i=(x_i-\min(x))/(\max(x)-\min(x))$$

Pythonic Implementation:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(data)
scaler.transform(data)
```

4.4 Splitting data into Input and Output Data

The input datas are the input features where as output data are the output label related to the features mapped.

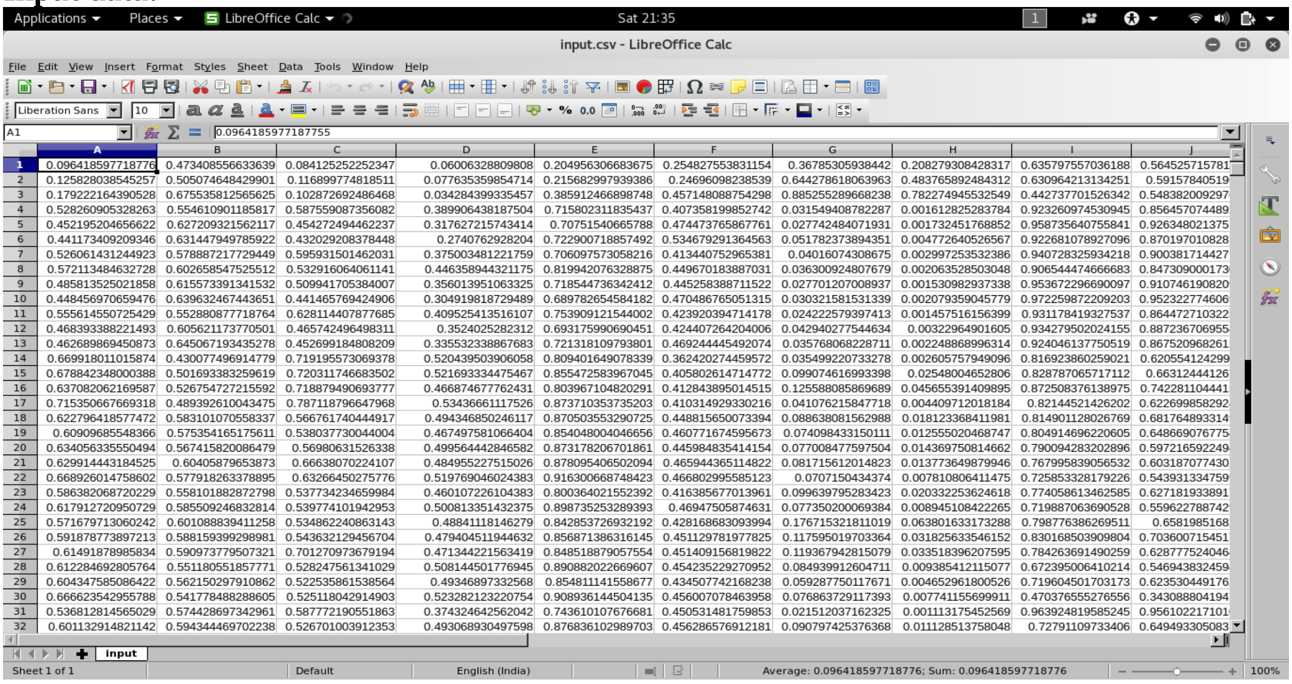
Using Numpy module to split a complete dataframe to Input and Output Dataset and save them.

Implementation:

```
#saving preprocessed data
@classmethod
def savedata(self,x, y):
    np.savetxt("PreProcessed_Data/input.csv",x,delimiter=",")
    np.savetxt("PreProcessed_Data/output.csv",y,delimiter=",")
```

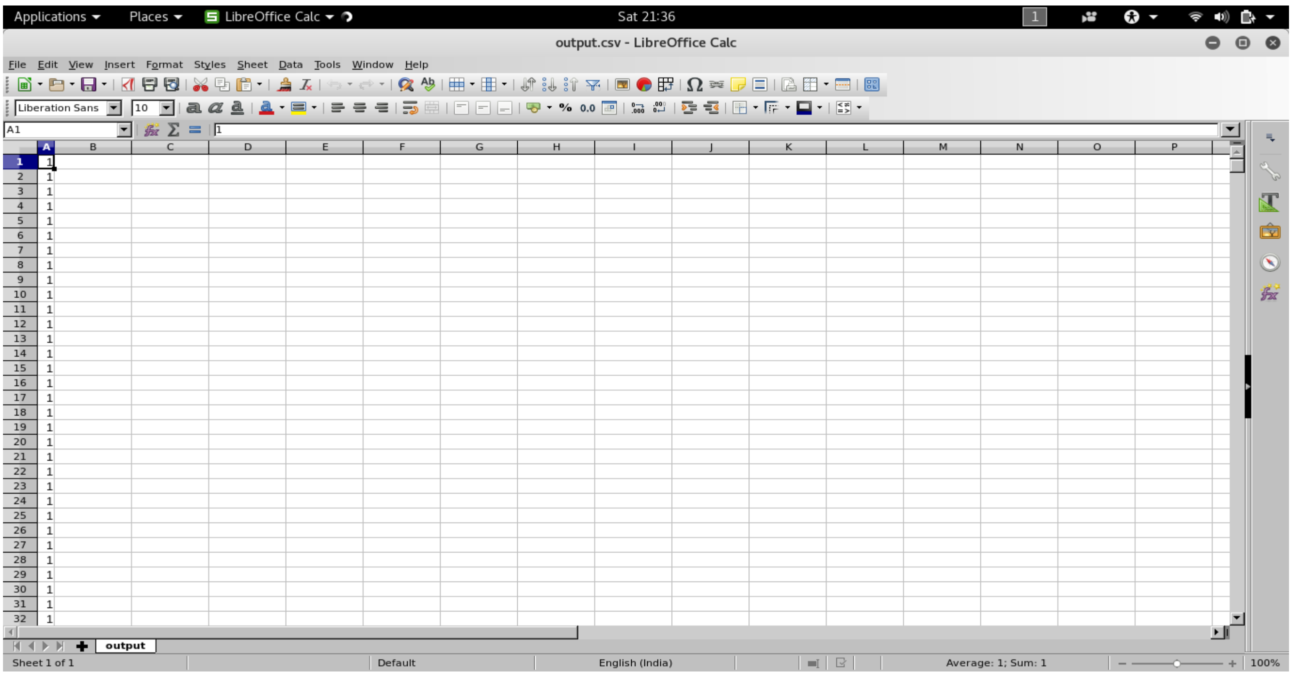
Snapshot of preprocessd dataset:

Input data:



	A	B	C	D	E	F	G	H	I	J
1	0.096418597718776	0.473408556633639	0.084125252252347	0.06006328809808	0.204956306683675	0.254827553831154	0.36785305938442	0.208279308428317	0.635797557036188	0.564525715781
2	0.125828038545257	0.505074648429901	0.116899774818511	0.077635359854714	0.215682997939386	0.24696098238539	0.644278618063963	0.483765892484312	0.630964213134251	0.59157840519
3	0.179222164390528	0.675535812565625	0.102872692486468	0.034284399335457	0.385912466898748	0.457148088754298	0.885255289668238	0.782274945532549	0.442737701526342	0.548382009297
4	0.528260905328263	0.554610901185817	0.587559087356082	0.389906438187504	0.715802311835437	0.407358199852742	0.031549408782287	0.001612825283784	0.923260974530945	0.856457074489
5	0.452195204656622	0.627209321562117	0.454272494462237	0.317627215743414	0.70751540665788	0.474473765867761	0.027742484071931	0.001732451768852	0.958735640755841	0.926348021375
6	0.441173409209346	0.631447949785922	0.432029208378448	0.2740762928204	0.722900718857492	0.534679291364563	0.051782373894351	0.004772640526567	0.922681078927096	0.870197010828
7	0.526061431244923	0.578887217729449	0.595931501462031	0.375003481221759	0.706097573058216	0.413440752965381	0.04016074308675	0.002997253532386	0.940728325934218	0.900381714427
8	0.572113484632728	0.602658547525512	0.532916064061141	0.446358944321175	0.819942076328875	0.449670183887031	0.036300924807679	0.002063528503048	0.906544474666683	0.847309000173
9	0.485813525021858	0.615573391341532	0.509941705384007	0.356013951063325	0.718544736342412	0.445258388711522	0.027701207008937	0.001530982937338	0.953672296690097	0.910746190820
10	0.448456970659476	0.639632467443651	0.441465769424906	0.304919818729489	0.689782654584182	0.470486765051315	0.030321581531339	0.002079359045779	0.972259872209203	0.952322774606
11	0.555614550725429	0.552880877718764	0.628114407877685	0.409525413516107	0.753909121544002	0.423920394714178	0.024222579397413	0.001457516156399	0.931178419327537	0.864472710322
12	0.468393388221493	0.605621173770501	0.465742496498311	0.3524025282312	0.693175990690451	0.424407264204006	0.042940277544634	0.00322964901605	0.934279502024155	0.887236706955
13	0.462689869450873	0.645067193435278	0.452699184808209	0.335532338867683	0.721318109793801	0.469244445492074	0.035768068228711	0.002248868996314	0.924046137750519	0.867520968261
14	0.669918011015874	0.430077496914779	0.719195573069378	0.520439503906058	0.809401649078339	0.362420274459572	0.035499220733278	0.002605757949096	0.816923860259021	0.620554124299
15	0.678842348000388	0.501693383259619	0.720311746683502	0.521693334475467	0.855472583967045	0.405802614714772	0.099074616993398	0.02548004652806	0.828787065717112	0.66312444126
16	0.637082062169587	0.526754727215592	0.718879490693777	0.466874677762431	0.803967104820291	0.412843895014515	0.125588085869689	0.045655391409895	0.872508376138975	0.742281104441
17	0.715350667669318	0.489392610043475	0.787118796647968	0.53436661117526	0.873710353735203	0.410314929330216	0.041076215847718	0.004409712018184	0.82144521426202	0.622699858292
18	0.622796418577472	0.583101070558337	0.566761740444917	0.494346850246117	0.870503553290725	0.448815650073394	0.088638081562988	0.018123368411981	0.814901128026769	0.681764893314
19	0.60909685548366	0.575354165175611	0.538037730044004	0.467497581066404	0.854048004046656	0.460771674595673	0.074098433150111	0.012555020468747	0.804914696220605	0.648669076775
20	0.634056335550494	0.567415820086479	0.56980631526338	0.499564442846582	0.873178206701861	0.445984835414154	0.077008477597504	0.014369750814662	0.790094283202896	0.597216592249
21	0.629914443184525	0.60405879653873	0.66638070224107	0.484955227515026	0.878095406502094	0.465944365114822	0.081715612014823	0.013773649879946	0.767995839056532	0.603187077430
22	0.668926014758602	0.577918263378895	0.63266450275776	0.519769046024383	0.916300668748423	0.466802995585123	0.0707150434374	0.007810806411475	0.725853328179226	0.543931334759
23	0.586382068720229	0.558101882872798	0.537734234659984	0.460107226104383	0.800364021552392	0.416385677013961	0.099639795283423	0.020332253624618	0.774058613462585	0.627181933891
24	0.617912720950729	0.585509246832814	0.539774101942953	0.500813351432375	0.898735253289393	0.46947505874631	0.077350200069384	0.008945108422265	0.719887063690528	0.559622788742
25	0.571679713060242	0.601088839411258	0.534862240863143	0.48841118146279	0.842853726932192	0.428168683093994	0.176715321811019	0.063801633173288	0.798776386269511	0.6581985168
26	0.591878773897213	0.588159399298981	0.543632129456704	0.479404511944632	0.856871386316145	0.451129781977825	0.117595019703364	0.031825633546152	0.830168503909804	0.703600715451
27	0.61491878985834	0.590973779507321	0.701270973679194	0.471344221563419	0.848518879057554	0.451409156819822	0.119367942815079	0.033518396207595	0.784263691490259	0.628777524046
28	0.612284692805764	0.551180551857771	0.528247561341029	0.508144501776945	0.890882022669607	0.454235229270952	0.084939912604711	0.009385412115077	0.672395006410214	0.546943832459
29	0.604347585086422	0.562150297910862	0.522535861538564	0.49346897332568	0.85481114155867	0.434507742168238	0.059287750117671	0.004652961800526	0.719604501703173	0.623530449176
30	0.666623542955788	0.541778488288605	0.525118042914903	0.523282123220754	0.908936144504135	0.456007078463958	0.076863729117393	0.007741155699911	0.470376555276556	0.343088804194
31	0.536812814565029	0.574428697342961	0.587772190551863	0.374324642562042	0.743610107676681	0.450531481759853	0.021512037162325	0.001113175452569	0.963924819585245	0.956102217101
32	0.601132914821142	0.594344469702238	0.526701003912353	0.493068930497598	0.876836102989703	0.456286576912181	0.090797425376368	0.011128513758048	0.72791109733406	0.649493305083

Output:



	A
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
30	1
31	1
32	1

5. Dataset Visualisation

Comparing each attributes of the male and female audio data to get a good idea of variation of voices among them , to compare i will be using histogram plotting to visualize.

5.1 Visualisation code:

```
#visualization
@classmethod
def plot_viz(self):
    df = self.df
    maledf = df.loc[df['label'] == "male"]
    femaledf = df.loc[df['label'] == "female"]
    name = df.columns

    for i in name[:-1]:

        plt.subplot(211)
        plt.hist(maledf[i],label = "male", color = "green")
        plt.legend(loc = "upper right")
        plt.subplot(212)
        plt.hist(femaledf[i],label="female", color= "orange")
        plt.legend(loc = "upper right")
        plt.suptitle("{} variation of male and female".format(i))
        plt.savefig("Figures/"+str(i)+".png")
```

5.2 Graphs:

Histogram Plotting

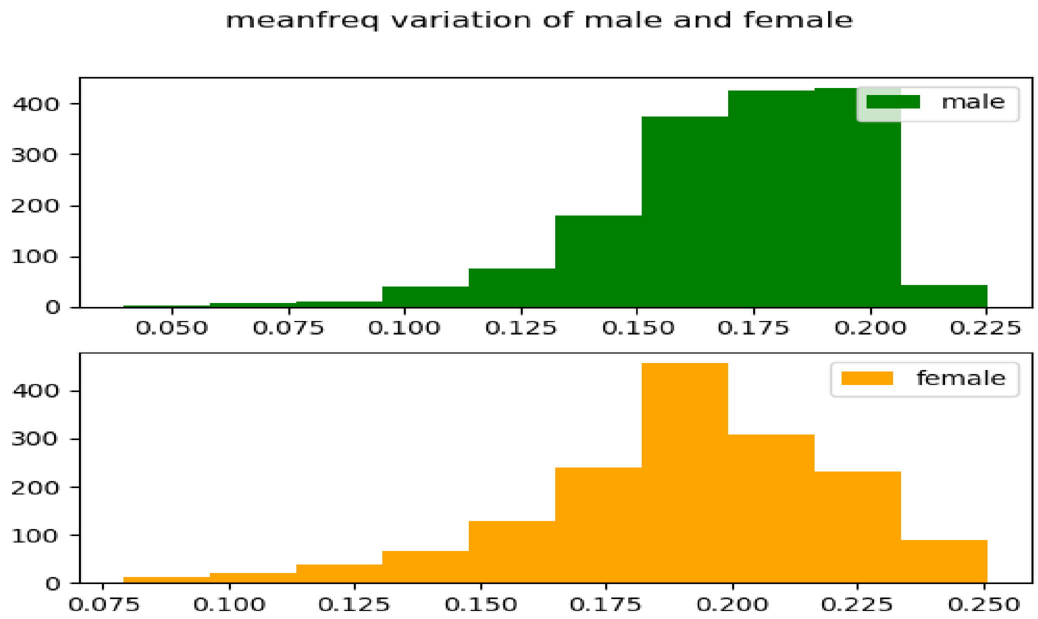


Illustration 1: Mean Frequency variation

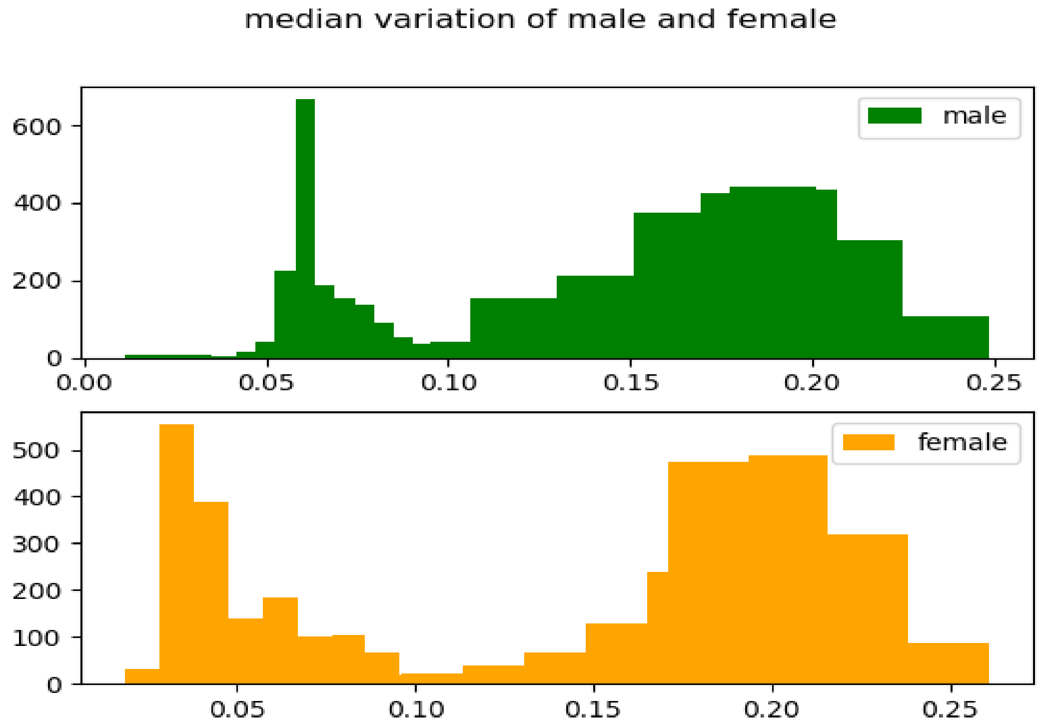


Illustration 2: Median Variation

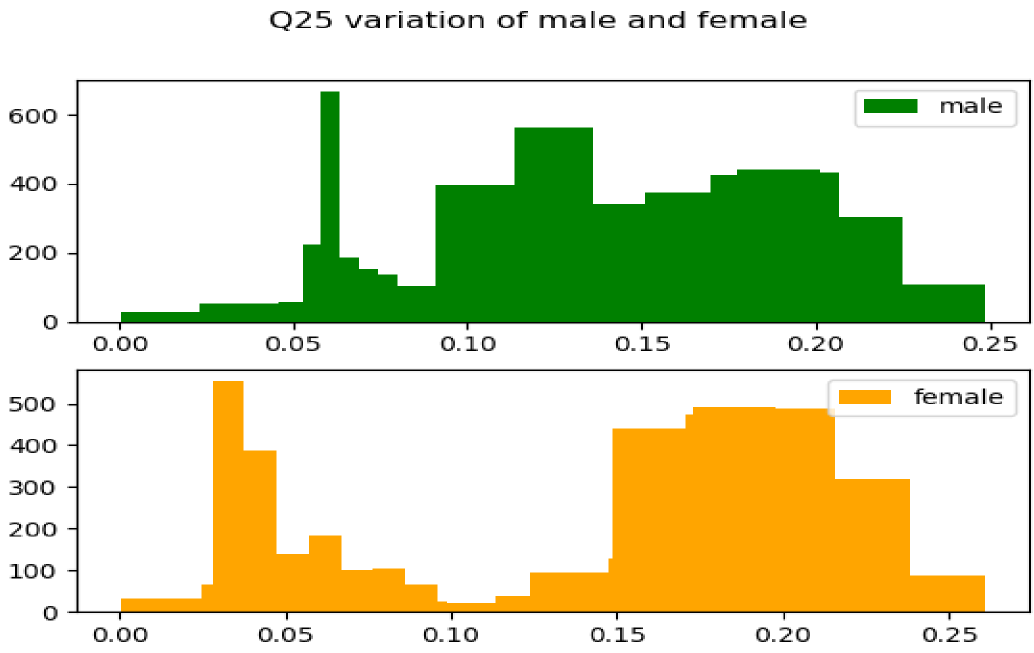


Illustration 3: First Quartile

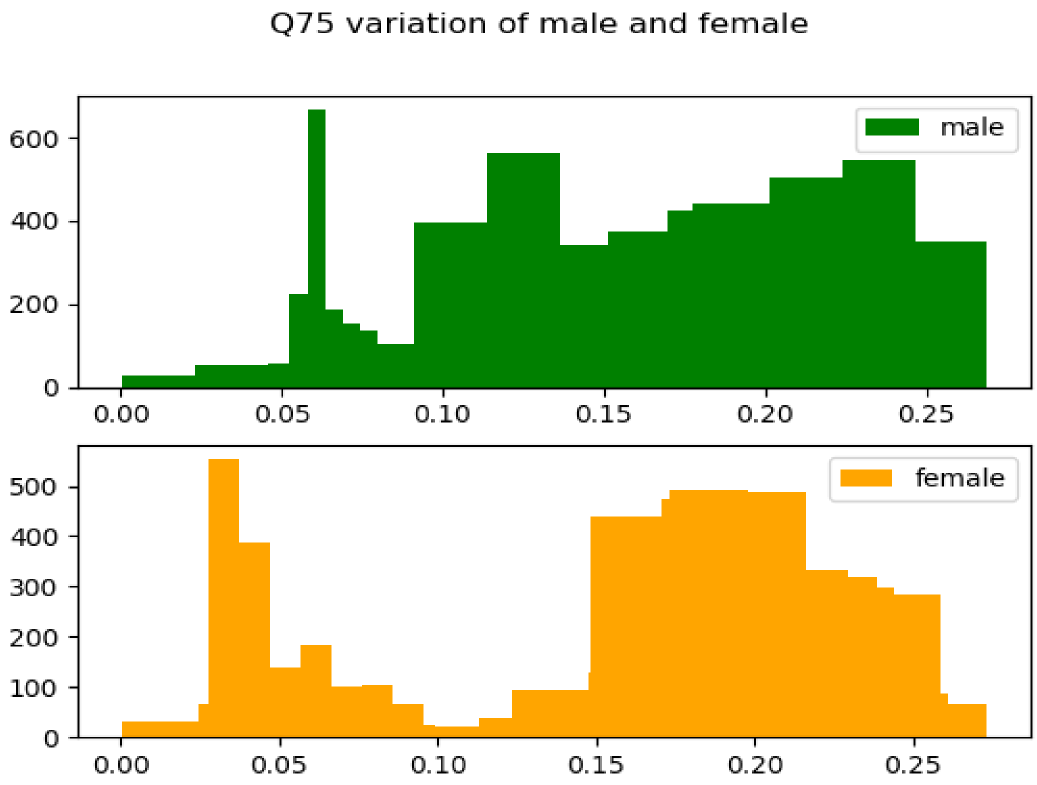


Illustration 4: Third Quartile

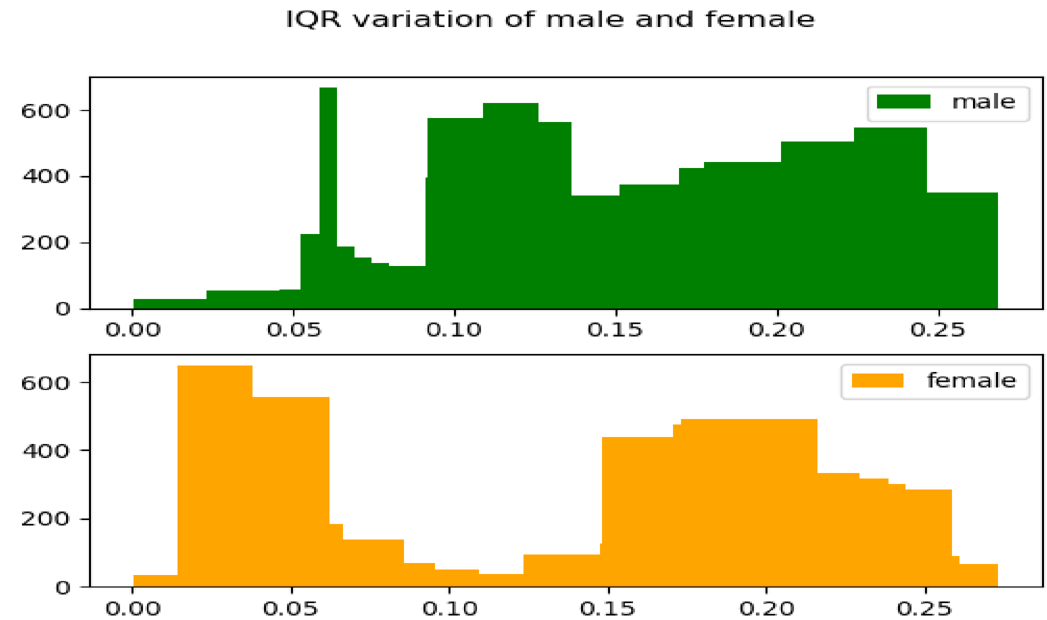


Illustration 5: Inter Quartile Variation

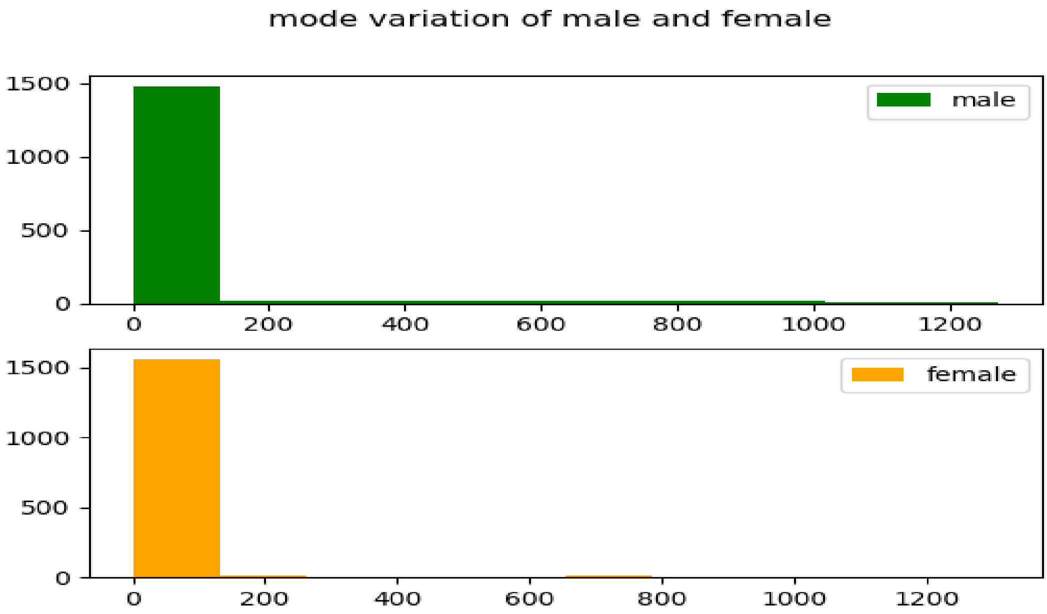


Illustration 6: Mode Variation

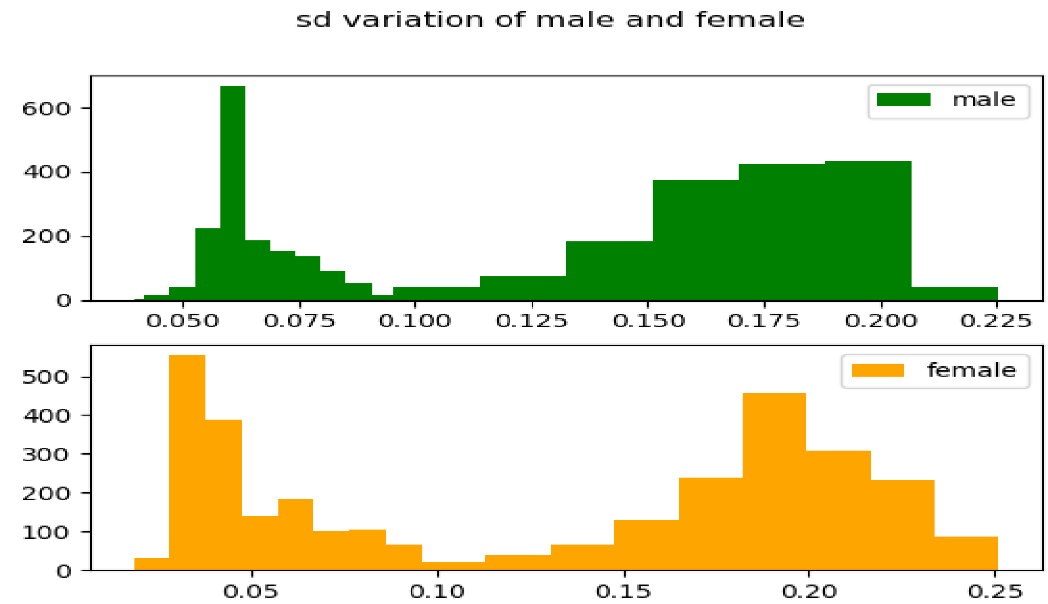


Illustration 7: Standard Deviation Variation

Illustration 8: Skewness Variation

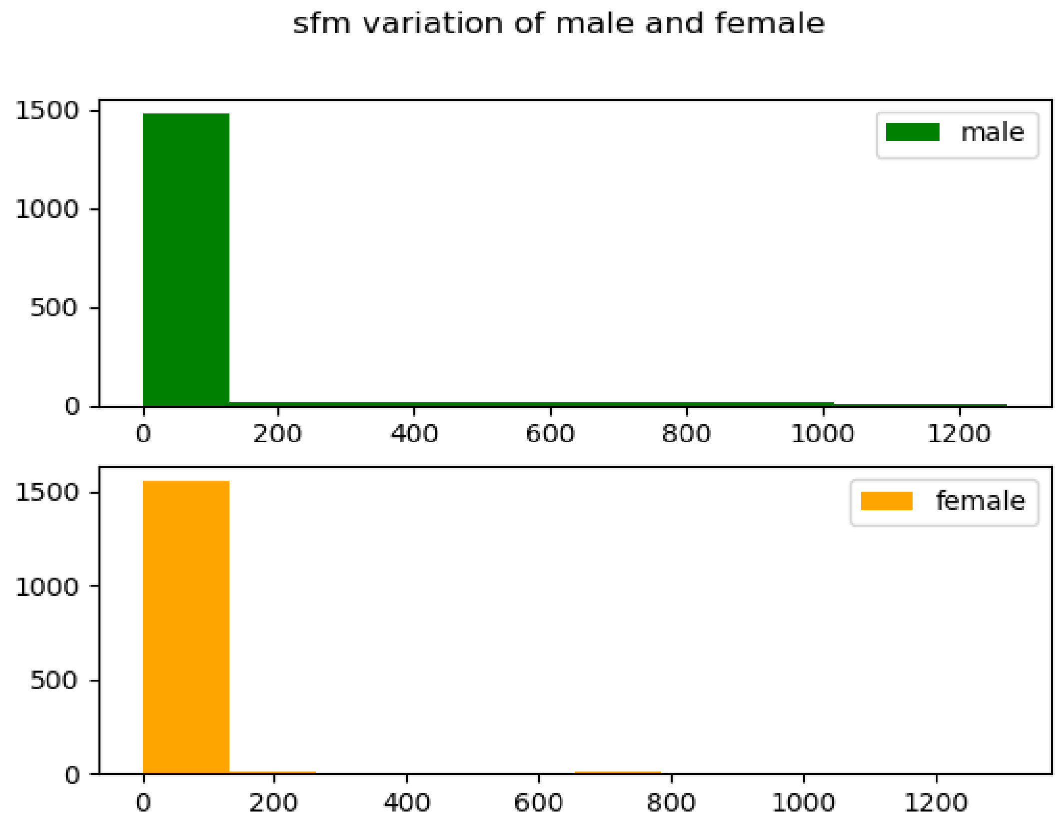


Illustration 9: Spectral Flatness

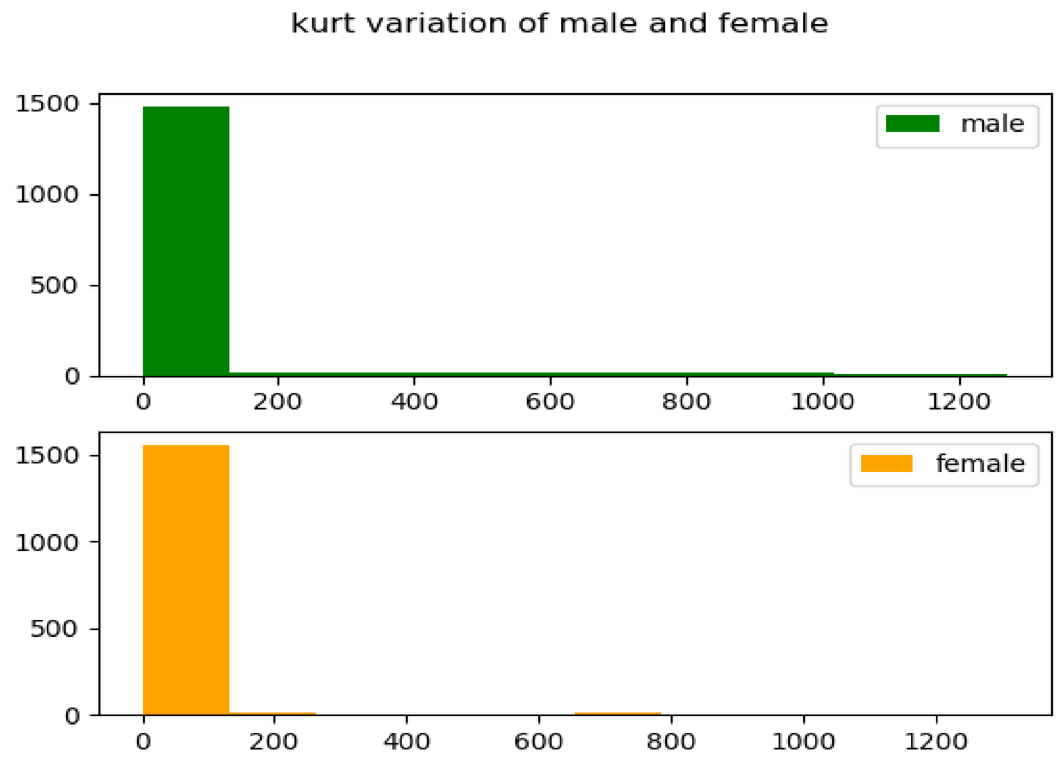


Illustration 10: Kurtosis Variation

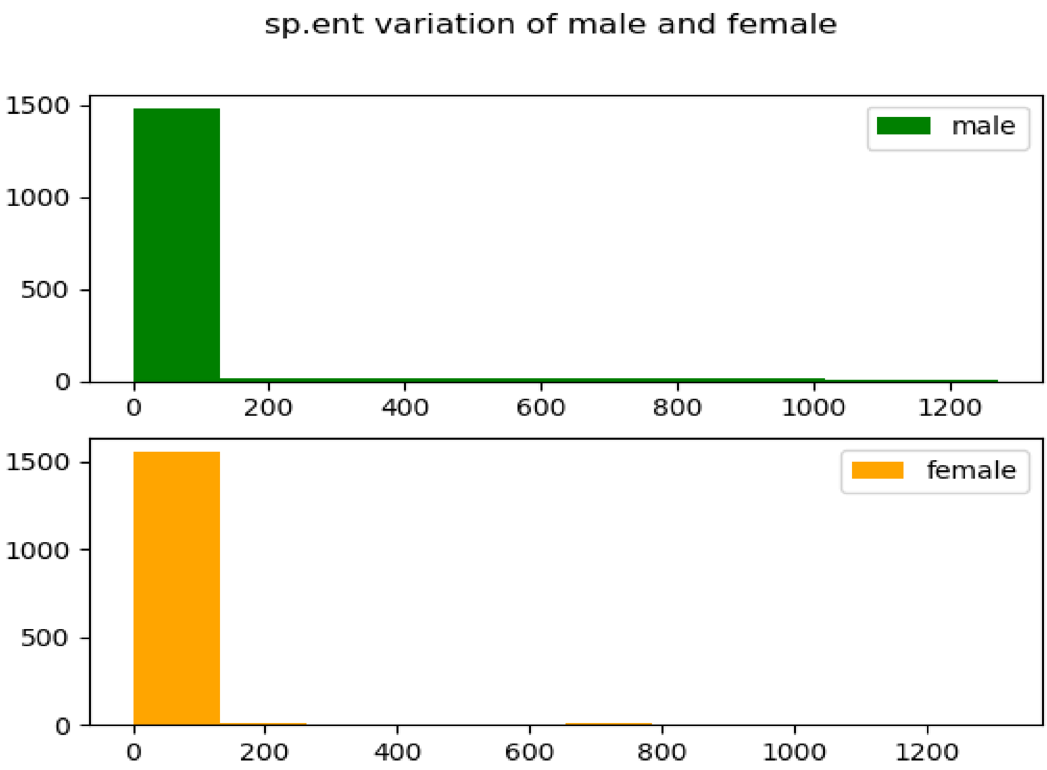


Illustration 11: Spectral Entropy

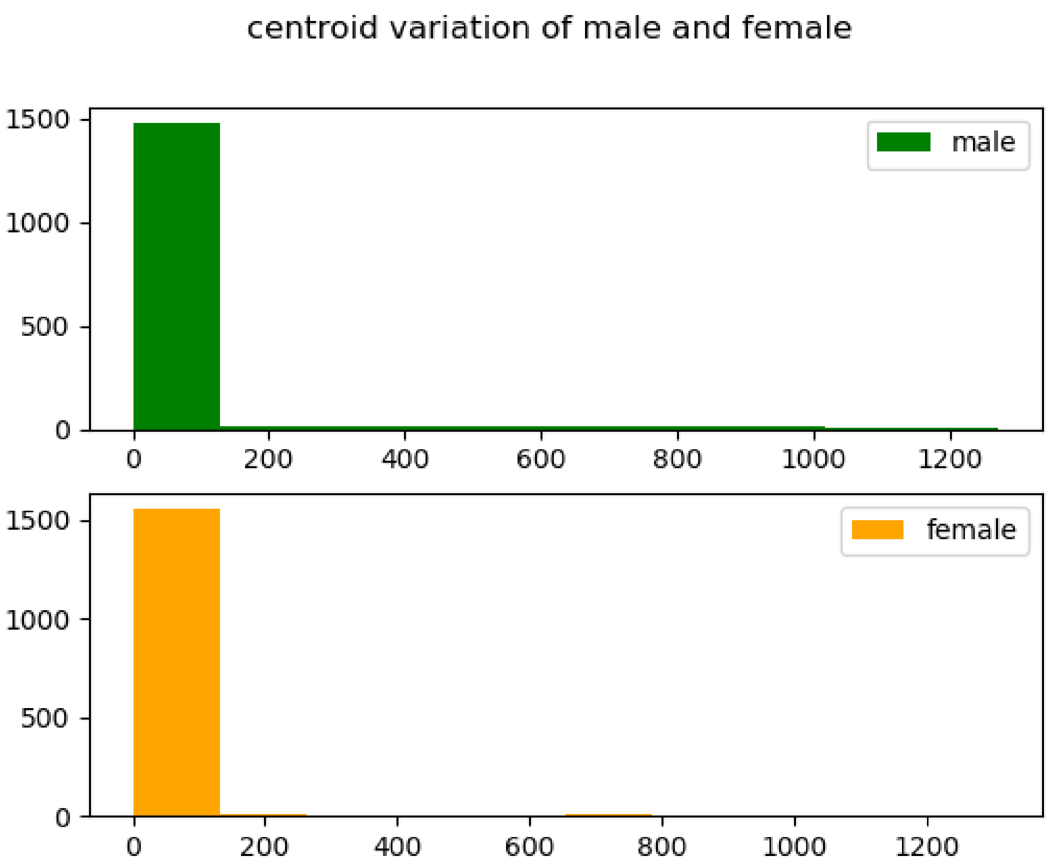


Illustration 12: Centroid Variation