In [1]:
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
a=pd.read_csv("C:\\Users\\reshma_koduri\\OneDrive\\Documents\\archive (2)\\voice.csv
a
```

Out[2]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955
...	...	...	...	...	...	...	...	...	...
3163	0.131884	0.084734	0.153707	0.049285	0.201144	0.151859	1.762129	6.630383	0.962934
3164	0.116221	0.089221	0.076758	0.042718	0.204911	0.162193	0.693730	2.503954	0.960716
3165	0.142056	0.095798	0.183731	0.033424	0.224360	0.190936	1.876502	6.604509	0.946854
3166	0.143659	0.090628	0.184976	0.043508	0.219943	0.176435	1.591065	5.388298	0.950436
3167	0.165509	0.092884	0.183044	0.070072	0.250827	0.180756	1.705029	5.769115	0.938829

3168 rows × 21 columns

In [3]:
```
a.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]:
```
a.describe()
```

Out[4]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	
count	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3
mean	0.180907	0.057126	0.185621	0.140456	0.224765	0.084309	3.140168	
std	0.029918	0.016652	0.036360	0.048680	0.023639	0.042783	4.240529	
min	0.039363	0.018363	0.010975	0.000229	0.042946	0.014558	0.141735	
25%	0.163662	0.041954	0.169593	0.111087	0.208747	0.042560	1.649569	
50%	0.184838	0.059155	0.190032	0.140286	0.225684	0.094280	2.197101	
75%	0.199146	0.067020	0.210618	0.175939	0.243660	0.114175	2.931694	
max	0.251124	0.115273	0.261224	0.247347	0.273469	0.252225	34.725453	1

In [5]:
```
list(a)
```

Out[5]:
```
['meanfreq',
 'sd',
 'median',
 'Q25',
 'Q75',
 'IQR',
 'skew',
 'kurt',
 'sp.ent',
 'sfm',
 'mode',
 'centroid',
 'meanfun',
 'minfun',
 'maxfun',
 'meandom',
 'mindom',
 'maxdom',
 'dfrange',
 'modindx',
 'label']
```

In [6]:
```
a.isna().sum()
```

Out[6]:
```
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 [7]:
```
b=a.drop(['Q25','Q75','IQR','kurt','sp.ent','sfm','centroid','skew','sd','modindx'],
b
```

Out[7]:

	meanfreq	median	mode	meanfun	minfun	maxfun	meandom	mindom	maxdom	d
0	0.059781	0.032027	0.000000	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.
1	0.066009	0.040229	0.000000	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.
2	0.077316	0.036718	0.000000	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.
3	0.151228	0.158011	0.083878	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.
4	0.135120	0.124656	0.104261	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.
...	...	...	...	...	...	...	...	...	...	
3163	0.131884	0.153707	0.200836	0.182790	0.083770	0.262295	0.832899	0.007812	4.210938	4.
3164	0.116221	0.076758	0.013683	0.188980	0.034409	0.275862	0.909856	0.039062	3.679688	3.
3165	0.142056	0.183731	0.008006	0.209918	0.039506	0.275862	0.494271	0.007812	2.937500	2.
3166	0.143659	0.184976	0.212202	0.172375	0.034483	0.250000	0.791360	0.007812	3.593750	3.
3167	0.165509	0.183044	0.267702	0.185607	0.062257	0.271186	0.227022	0.007812	0.554688	0.

3168 rows × 11 columns

In [8]:
```
c=pd.get_dummies(b,dtype=int)
c
```

Out[8]:

	meanfreq	median	mode	meanfun	minfun	maxfun	meandom	mindom	maxdom	d
0	0.059781	0.032027	0.000000	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.
1	0.066009	0.040229	0.000000	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.
2	0.077316	0.036718	0.000000	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.
3	0.151228	0.158011	0.083878	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.
4	0.135120	0.124656	0.104261	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.
...	...	...	...	...	...	...	...	...	...	
3163	0.131884	0.153707	0.200836	0.182790	0.083770	0.262295	0.832899	0.007812	4.210938	4.
3164	0.116221	0.076758	0.013683	0.188980	0.034409	0.275862	0.909856	0.039062	3.679688	3.
3165	0.142056	0.183731	0.008006	0.209918	0.039506	0.275862	0.494271	0.007812	2.937500	2.

	meanfreq	median	mode	meanfun	minfun	maxfun	meandom	mindom	maxdom	d
3166	0.143659	0.184976	0.212202	0.172375	0.034483	0.250000	0.791360	0.007812	3.593750	3.
3167	0.165509	0.183044	0.267702	0.185607	0.062257	0.271186	0.227022	0.007812	0.554688	0.

3168 rows × 12 columns

In [9]:
```
y=c['meanfreq']
y
```

Out[9]:
```
0       0.059781
1       0.066009
2       0.077316
3       0.151228
4       0.135120
          ...
3163    0.131884
3164    0.116221
3165    0.142056
3166    0.143659
3167    0.165509
Name: meanfreq, Length: 3168, dtype: float64
```

In [10]:
```
x=c.drop(['meanfreq'],axis=1)
x
```

Out[10]:

	median	mode	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	lab
0	0.032027	0.000000	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	
1	0.040229	0.000000	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	
2	0.036718	0.000000	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	
3	0.158011	0.083878	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	
4	0.124656	0.104261	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	
...	...	...	...	...	...	...	...	...	...	
3163	0.153707	0.200836	0.182790	0.083770	0.262295	0.832899	0.007812	4.210938	4.203125	
3164	0.076758	0.013683	0.188980	0.034409	0.275862	0.909856	0.039062	3.679688	3.640625	
3165	0.183731	0.008006	0.209918	0.039506	0.275862	0.494271	0.007812	2.937500	2.929688	
3166	0.184976	0.212202	0.172375	0.034483	0.250000	0.791360	0.007812	3.593750	3.585938	
3167	0.183044	0.267702	0.185607	0.062257	0.271186	0.227022	0.007812	0.554688	0.546875	

3168 rows × 11 columns

In [11]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [12]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
```

```
n_estimators=[25,50,75,100,125,150,175,200]
criterion=['mse']
max_depth=[3,5,10]
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth
rfc_reg = GridSearchCV(reg, parameters)
rfc_reg.fit(x_train,y_train)
```

Out[12]:
```
GridSearchCV(estimator=RandomForestRegressor(),
             param_grid={'criterion': ['mse'], 'max_depth': [3, 5, 10],
                         'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]})
```

In [13]:
```
rfc_reg.best_params_
```

Out[13]:
```
{'criterion': 'mse', 'max_depth': 10, 'n_estimators': 150}
```

In [23]:
```
reg=RandomForestRegressor(n_estimators=150,criterion='mse',max_depth=10)
```

In [24]:
```
reg.fit(x_train,y_train)
```

Out[24]:
```
RandomForestRegressor(max_depth=10, n_estimators=150)
```

In [31]:
```
y_pred=reg.predict(x_test)
y_pred
```

Out[31]:
```
array([0.17568116, 0.1906409 , 0.15965559, ..., 0.22959718, 0.21601968,
       0.18506223])
```

In [32]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[32]:
```
0.9477014545373129
```

In [37]:
```
Results= pd.DataFrame(columns=['actual','Predicted'])
Results['actual']=y_test
Results['Predicted']=y_pred
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

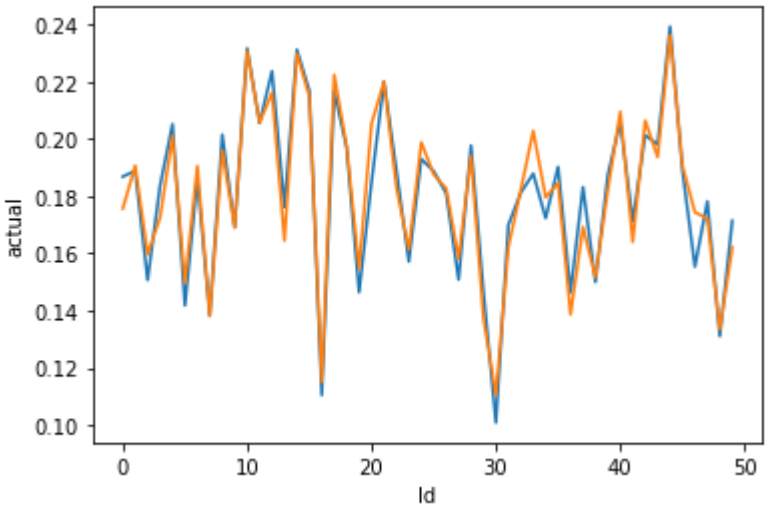
Out[37]:

	index	actual	Predicted	Id
0	2148	0.186833	0.175681	0
1	1124	0.188879	0.190641	1
2	170	0.150705	0.159656	2
3	3158	0.183667	0.172573	3
4	2229	0.205159	0.201107	4
5	1960	0.141798	0.149672	5
6	411	0.186064	0.190510	6
7	457	0.138354	0.138139	7
8	2881	0.201499	0.196011	8

	index	actual	Predicted	Id
9	602	0.169100	0.169062	9

In [39]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='Id',y='actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[39]:
```
[]
```



In [ ]: