In [ ]:
```
from platform import python_version
python_version()
```

Out[ ]:
```
'3.9.6'
```

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from statistics import mean, stdev
import seaborn as sns
import ml
from filter import Filter
```

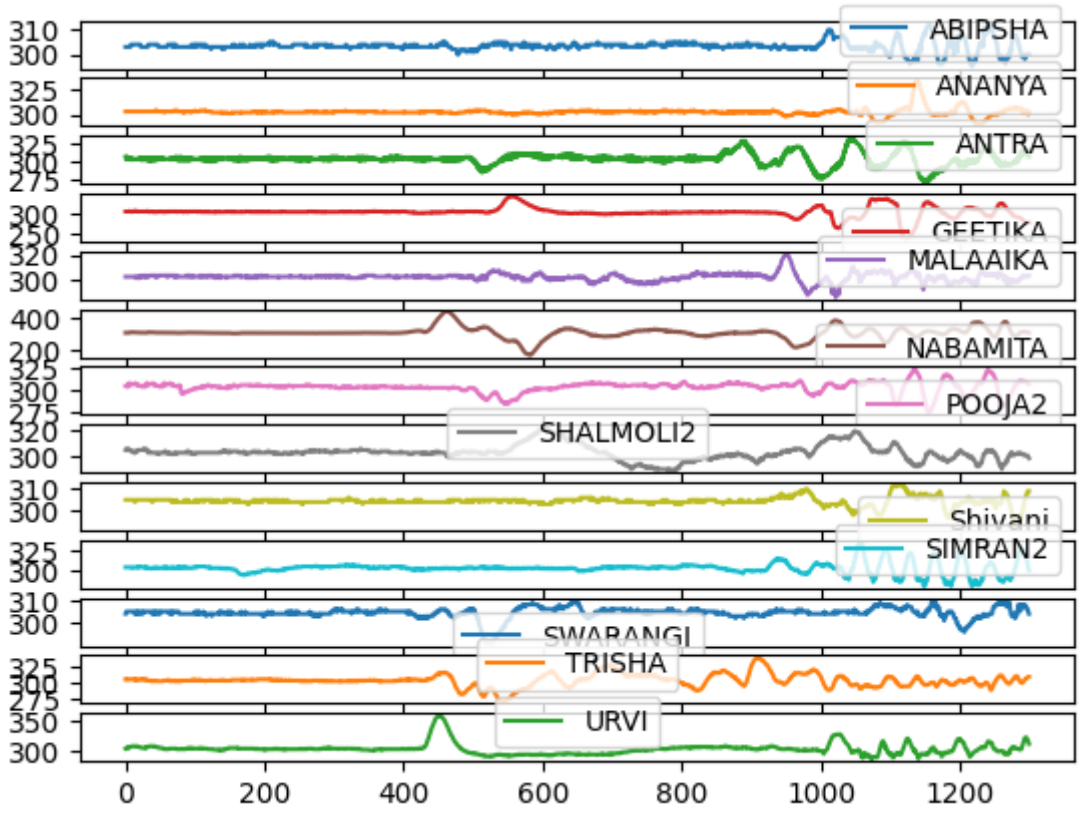
In [ ]:
```
ts = ['ABIPSHA',
      'ANANYA',
      'ANTRA',
      'GEETIKA',
      'MALAAIKA',
      'NABAMITA',
      'POOJA2',
      'SHALMOLI2',
      'Shivani',
      'SIMRAN2',
      'SWARANGI',
      'TRISHA',
      'URVI' ]

td = pd.concat(list(map(lambda n: pd.read_csv(n, header=None), ['data/' + s + '.csv
td = td.iloc[0:1300]
td.columns = ts
td.head()
```

Out[ ]:

	ABIPSHA	ANANYA	ANTRA	GEETIKA	MALAAIKA	NABAMITA	POOJA2	SHALMOLI2	Shivani
0	303.0	303	306.0	305	303.0	305	305	305	305
1	303.0	303	302.0	305	303.0	305	305	306	305
2	303.0	303	306.0	305	303.0	305	306	306	305
3	303.0	303	303.0	305	303.0	305	304	306	305
4	303.0	303	305.0	305	303.0	305	305	306	305

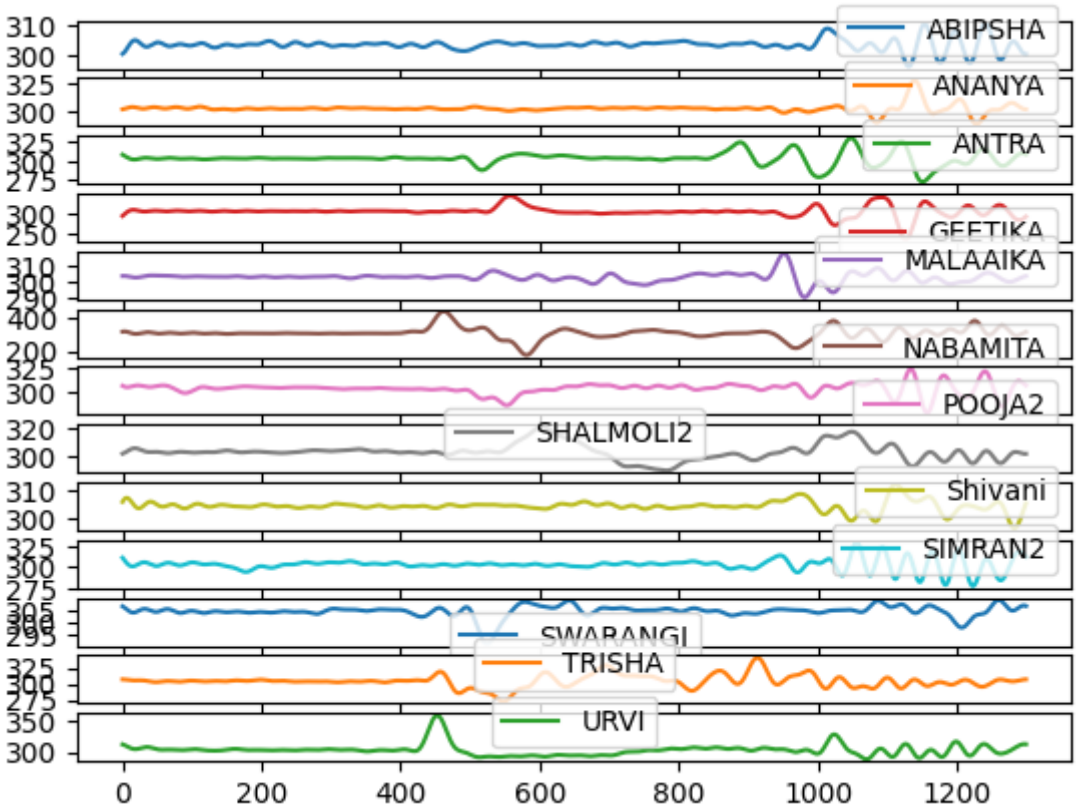
In [ ]:
```
td.plot(subplots=True)
```

Out[ ]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>], dtype=object)
```



In [ ]:
```
def lpf(sig, fq, cf):
    f = Filter(sig)
    f.cfft(fq)
    return f.af(cf)

for col in td:
    td[f'{col}'] = lpf(td[f'{col}'], 1000, 30)

td.plot(subplots=True)
```

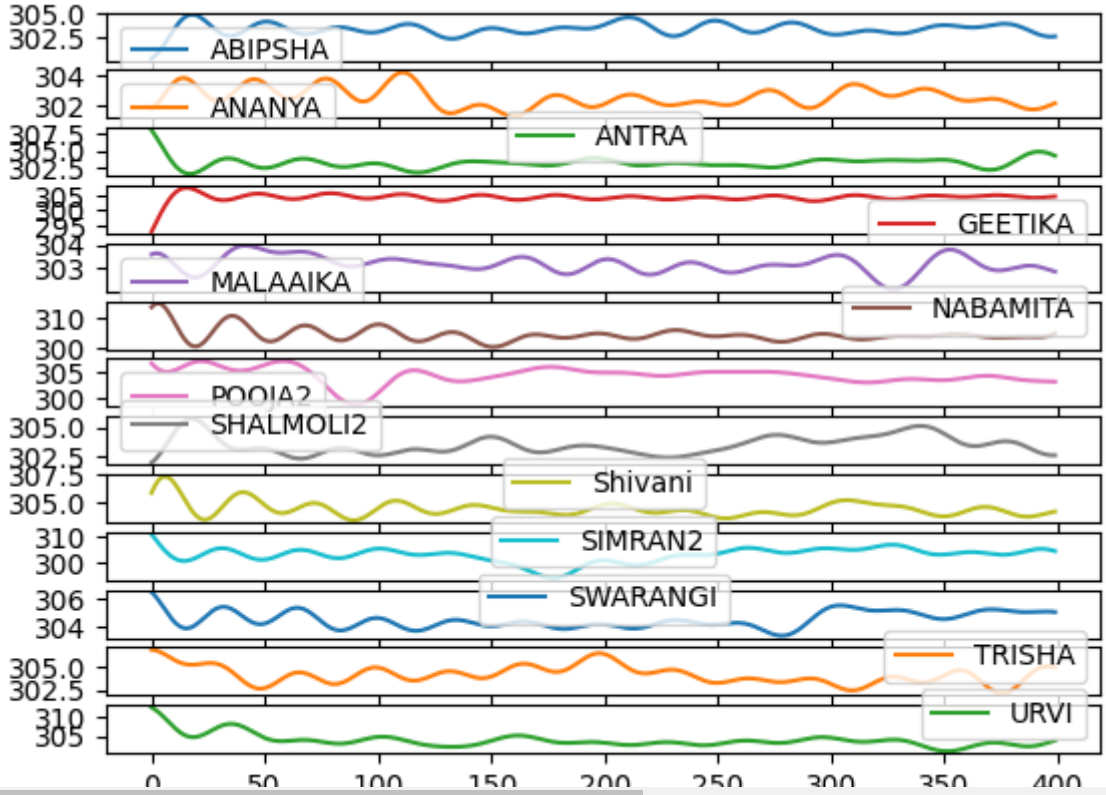
Out[ ]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>,
       <AxesSubplot:>], dtype=object)
```



In [ ]:
```
nu = td.iloc[0:400]
nu.plot(subplots=True)
nu.describe()
```

Out[ ]:

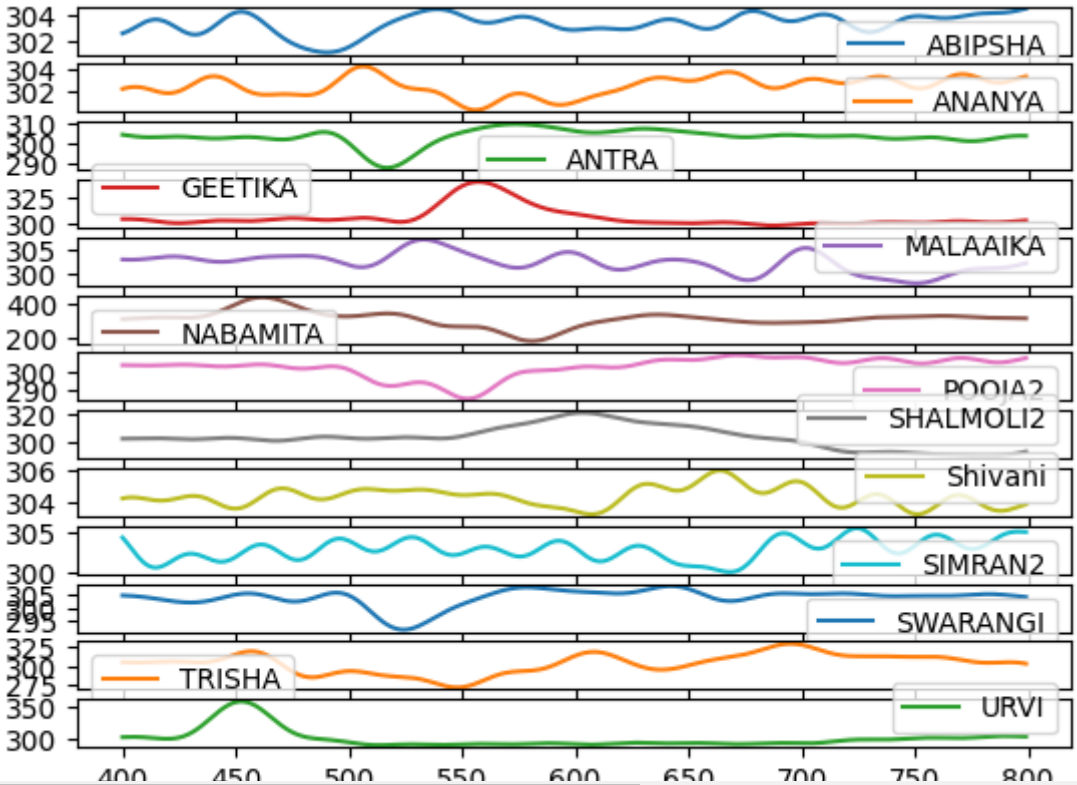
	ABIPSHA	ANANYA	ANTRA	GEETIKA	MALAAIKA	NABAMITA	POOJA2	SHAL
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.
mean	303.369158	302.549012	303.214701	304.373100	303.173593	304.345084	304.446932	303.
std	0.612069	0.643298	0.748663	1.247759	0.359020	2.294331	1.549197	0.
min	300.287272	301.260296	301.621403	292.989266	302.107746	300.204559	298.890839	302.
25%	303.000486	302.057043	302.837100	303.911900	302.968267	303.267016	303.667931	302.
50%	303.391855	302.446109	303.159551	304.470062	303.151090	303.935077	304.625275	303.
75%	303.751629	302.972896	303.588251	304.927557	303.393716	304.677028	305.260555	303.
max	304.891854	304.212679	308.061266	307.553088	303.966006	314.977256	307.193120	305.



In [ ]:
```
sm = td.iloc[400:800]
sm.plot(subplots=True)
sm.describe()
```

Out[ ]:

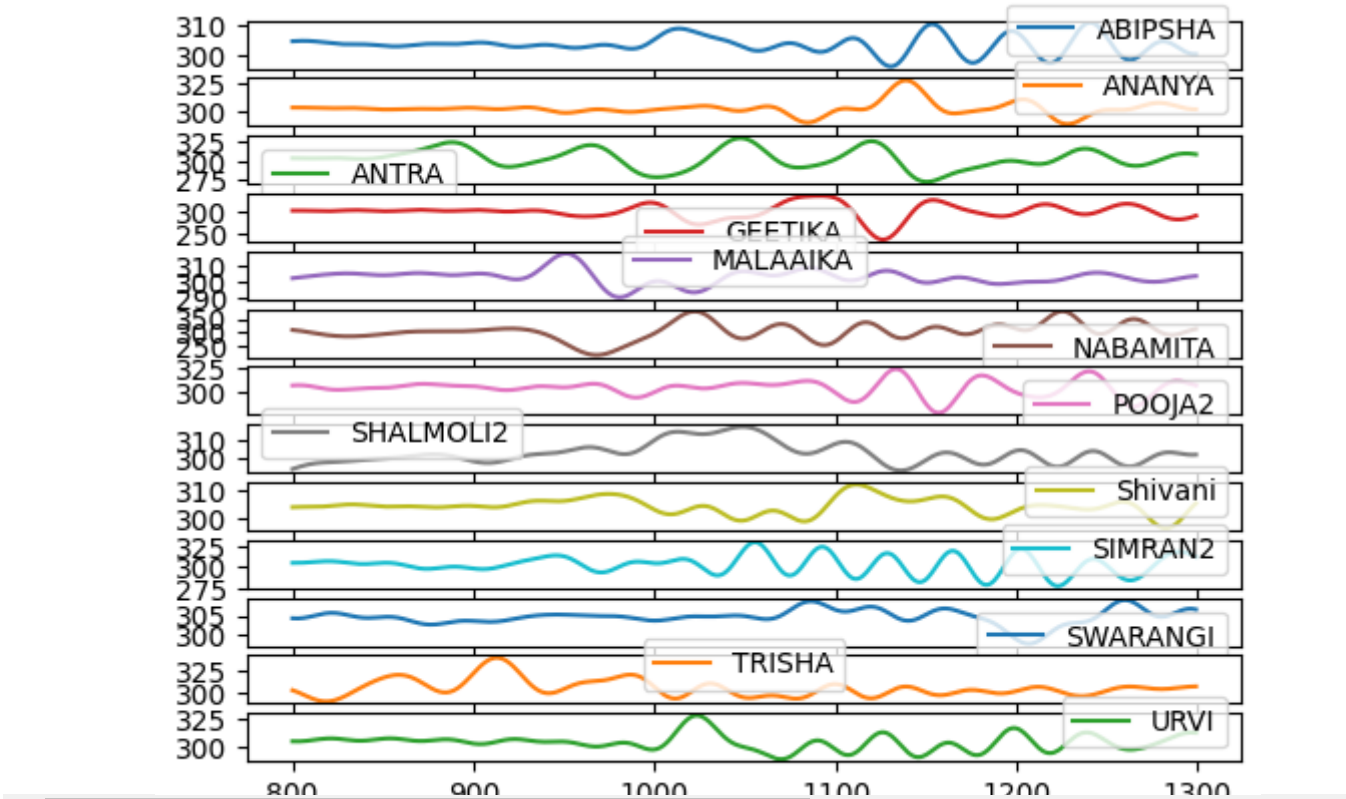
	ABIPSHA	ANANYA	ANTRA	GEETIKA	MALAAIKA	NABAMITA	POOJA2	SHAL
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.
mean	303.367352	302.418227	303.071836	306.183804	302.149597	307.098479	302.162695	303.
std	0.745821	0.843893	4.077342	9.953300	2.045894	48.311153	5.271551	7.
min	301.219879	300.306725	287.713577	298.259959	297.838750	175.303043	285.734674	290.
25%	302.977418	301.785610	302.383388	301.017169	300.961483	286.456927	301.444526	300.
50%	303.539805	302.515434	303.301247	302.570921	302.413754	312.859663	303.251399	302.
75%	303.890406	303.061380	305.241770	305.284718	303.411588	324.014052	305.923135	308.
max	304.483817	304.179985	309.385682	341.327905	307.019710	434.182494	308.507925	320.



In [ ]:
```
ch = td.iloc[800:1300]
ch.plot(subplots=True)
ch.describe()
```

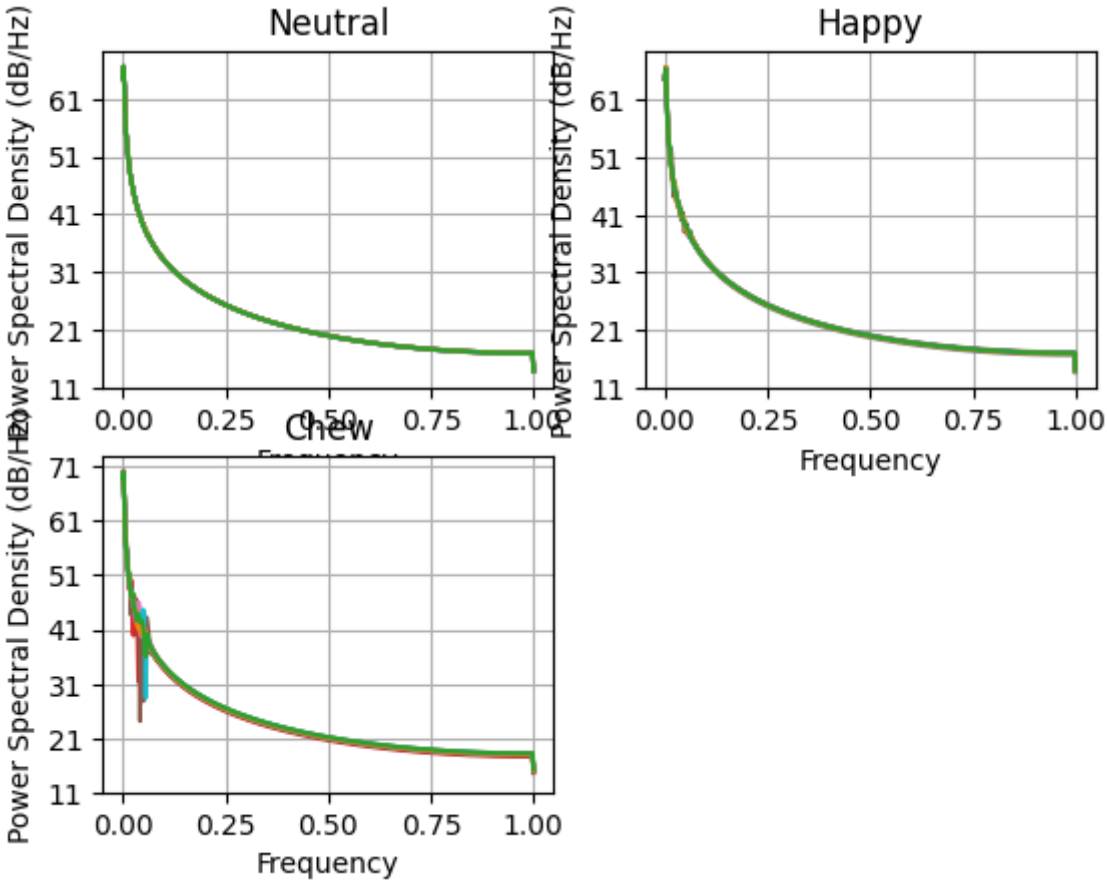
Out[ ]:

	ABIPSHA	ANANYA	ANTRA	GEETIKA	MALAAIKA	NABAMITA	POOJA2	SHAL
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.000000	500.
mean	303.354791	302.548209	302.638771	300.986476	302.871448	303.315150	304.380299	302.
std	2.871466	5.967325	13.026797	17.820802	4.549819	31.849874	8.297333	5.
min	296.056861	288.488706	272.235591	235.992880	290.036530	216.155823	277.696768	293.
25%	302.240609	300.662126	294.359633	292.646297	300.286335	288.775841	301.953086	298.
50%	303.311983	302.147890	303.010134	302.717942	303.440696	303.951650	305.623742	301.
75%	304.472585	303.345112	311.728407	309.013998	305.121311	319.109594	308.517189	304.
max	310.798957	327.697689	329.933999	337.732096	317.874735	377.856943	324.715989	317.



In [ ]:

```
mnuamp = []
msmamp = []
mchamp = []
for col in td:
    plt.subplot(221)
    plt.title("Neutral")
    nuamp, nufreq = plt.psd(nu[f'{col}'], 1000)
    plt.subplot(222)
    plt.title("Happy")
    smamp, sfreq = plt.psd(sm[f'{col}'], 1000)
    plt.subplot(223)
    plt.title("Chew")
    champ, chfreq = plt.psd(ch[f'{col}'], 1000)
    mnuamp.append(mean(nuamp))
    msmamp.append(mean(smamp))
    mchamp.append(mean(champ))
```

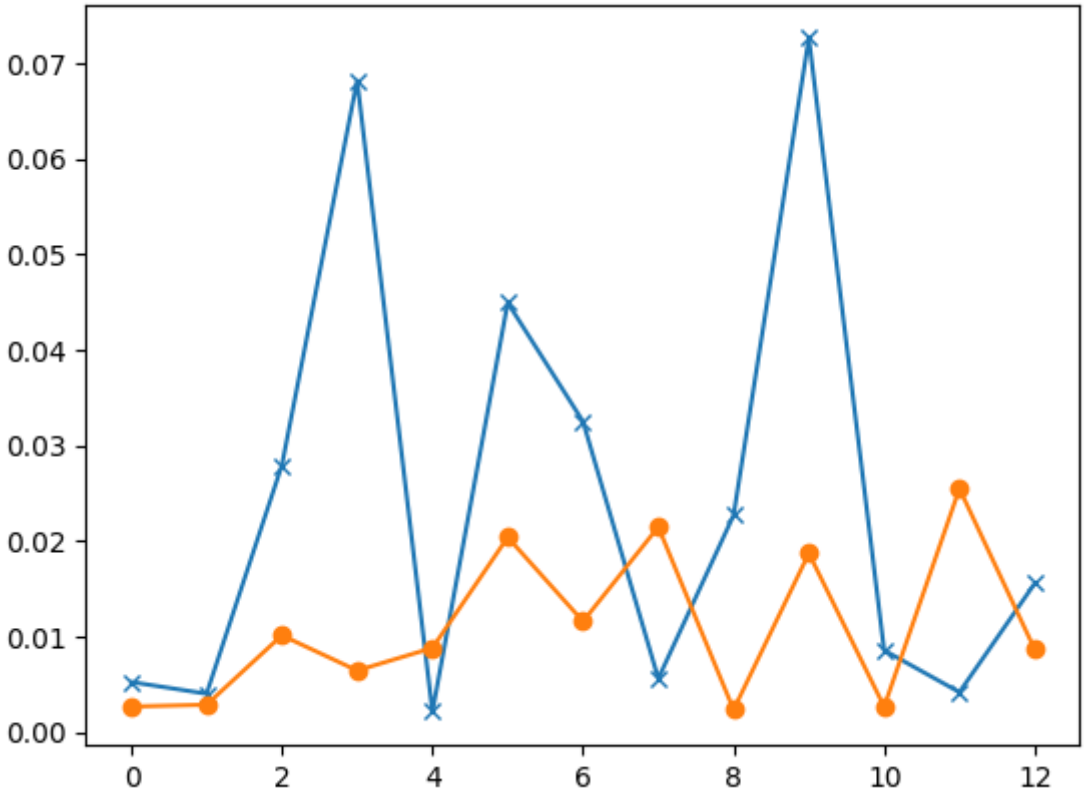


In [ ]:
```
fch = pd.DataFrame({'MPSD': mchamp, 'MEAN': ch.mean(), 'SD': ch.std(), 'MAX': ch.ma
fnu = pd.DataFrame({'MPSD': mnuamp, 'MEAN': nu.mean(), 'SD': nu.std(), 'MAX': ch.ma
fsm = pd.DataFrame({'MPSD': msmamp, 'MEAN': sm.mean(), 'SD': sm.std(), 'MAX': ch.ma
f = pd.concat([fnu, fsm, fch])
f.reset_index(drop=True, inplace=True)
```

In [ ]:
```
def dwt(sig):
    # return pywt.dwt(sig, wl)[1]
    return Filter(sig).wt()

wtnusd = []
wtnum = []
wtnumax = []
wtsmsd = []
wtsmm = []
wtsmmax = []
wtchsd = []
wtchm = []
wtchmax = []
for col in td:
    wtnumax.append(max(dwt(nu[f'{col}'])))
    wtsmmax.append(max(dwt(sm[f'{col}'])))
    wtchmax.append(max(dwt(ch[f'{col}'])))
    wtnum.append(mean(dwt(nu[f'{col}'])))
    wtsmm.append(mean(dwt(sm[f'{col}'])))
    wtchm.append(mean(dwt(ch[f'{col}'])))
    wtnusd.append(stdev(dwt(nu[f'{col}'])))
    wtsmsd.append(stdev(dwt(sm[f'{col}'])))
    wtchsd.append(stdev(dwt(ch[f'{col}'])))

wtnu = pd.DataFrame({'WTSD': wtnusd, 'WTM': wtnum, 'WTMAX': wtnumax})
wtsm = pd.DataFrame({'WTSD': wtsmsd, 'WTM': wtsmm, 'WTMAX': wtsmmax})
wtch = pd.DataFrame({'WTSD': wtchsd, 'WTM': wtchm, 'WTMAX': wtchmax})
```

```
plt.plot(wtnumax, marker='x')
plt.plot(wtsmmax, marker='o')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x26a8b0213d0>]
```



In [ ]:
```
f1 = pd.concat([wtnu, wtsm, wtch])
f1.reset_index(drop=True, inplace=True)
f = pd.concat([f1, f], join='outer', axis=1)
f.head()
```

Out[ ]:

	WTSD	WTM	WTMAX	MPSD	MEAN	SD	MAX	LABEL
0	0.000663	9.335592e-06	0.005211	22949.724993	303.369158	0.612069	310.798957	0
1	0.000598	7.467573e-06	0.003986	22804.334268	302.549012	0.643298	327.697689	0
2	0.002781	-1.000681e-05	0.027899	22949.973001	303.214701	0.748663	329.933999	0
3	0.008866	2.514819e-05	0.068186	23107.326456	304.373100	1.247759	337.732096	0
4	0.000376	-8.590971e-08	0.002104	22900.932021	303.173593	0.359020	317.874735	0

In [ ]:
```
for col in f:
    if(col!='LABEL'):
        f[f'{col}'] = list(map(lambda x: (x-f[f'{col}'].mean())/f[f'{col}'].std(),
```

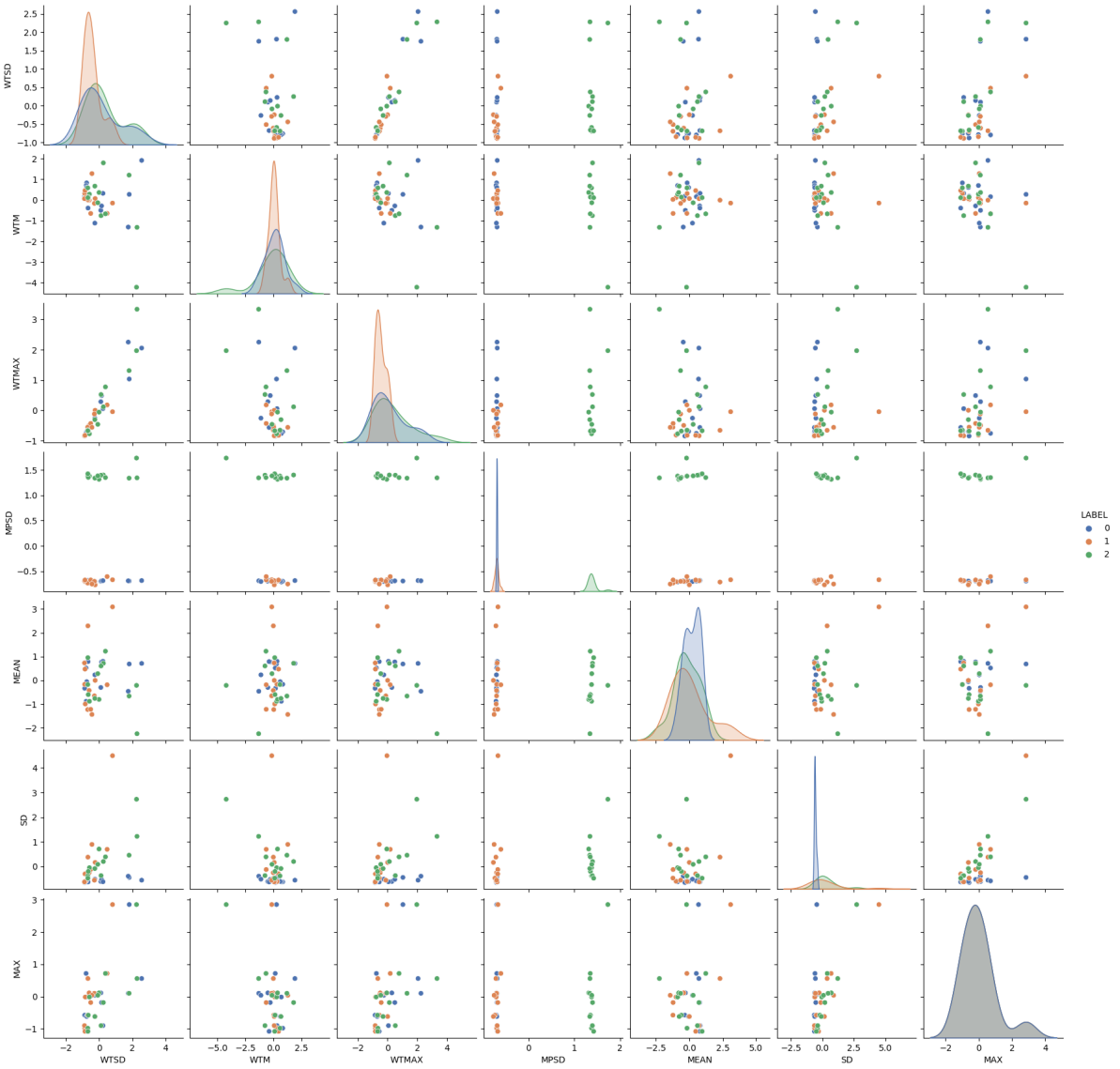
In [ ]:
```
f.head()
```

Out[ ]:

	WTSD	WTM	WTMAX	MPSD	MEAN	SD	MAX	LABEL
0	-0.762385	0.825287	-0.710513	-0.699056	-0.166129	-0.621475	-0.982101	0
1	-0.788629	0.696990	-0.764297	-0.712248	-0.882474	-0.618130	-0.016471	0
2	0.095018	-0.503162	0.285417	-0.699033	-0.301038	-0.606846	0.111316	0
3	2.558222	1.911306	2.053958	-0.684755	0.710751	-0.553392	0.556916	0
4	-0.878503	0.178212	-0.846896	-0.703483	-0.336943	-0.648576	-0.577776	0

In [ ]:
```
sns.pairplot(f, hue='LABEL', palette='deep')
```

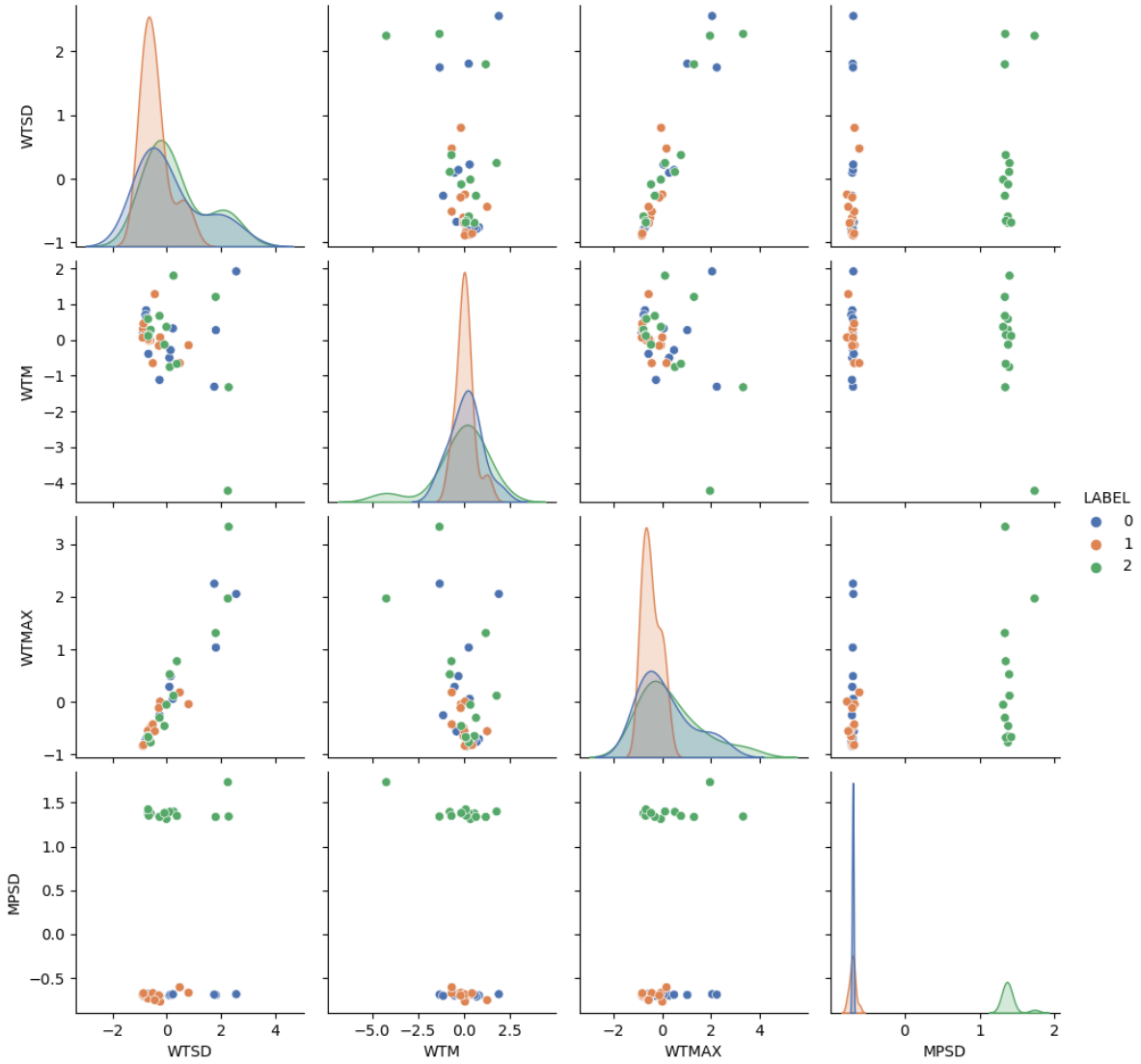
Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x26a8a8e8850>
```



In [ ]:
```
f = f.drop(['MAX', 'SD', 'MEAN'], axis=1)
```

In [ ]:
```
sns.pairplot(f, hue='LABEL', palette='deep')
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x26a8af9fbe0>
```



In [ ]:
```
f.head()
```

Out[ ]:

	WTSD	WTM	WTMAX	MPSD	LABEL
0	-0.762385	0.825287	-0.710513	-0.699056	0
1	-0.788629	0.696990	-0.764297	-0.712248	0
2	0.095018	-0.503162	0.285417	-0.699033	0
3	2.558222	1.911306	2.053958	-0.684755	0
4	-0.878503	0.178212	-0.846896	-0.703483	0

In [ ]:
```
nearest_neighbor = 1
m = ml.Knn(f, nearest_neighbor)
plt.title("Error vs Neighbours")
plt.plot(np.arange(1,32,1), m.op(), marker='o')
plt.xticks(np.arange(1,32,1))
m.predict()
```

```
Predictions
[2 2 1 1 2 2 0 2]
['chew', 'chew', 'smile', 'smile', 'chew', 'chew', 'neutral', 'chew']
Actual Values
33    2
36    2
4     0
13    1
30    2
26    2
6     0
27    2
Name: LABEL, dtype: int64
['chew', 'chew', 'neutral', 'smile', 'chew', 'chew', 'neutral', 'chew']
Accuracy
0.875
```

Out[ ]: