```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
from sklearn.decomposition import PCA

df1 = pd.read_csv("IMUeatingfile.csv")


# Variables storing the values for features from Phase 1
Ax_X_mean_noneating
Ax_X_std_noneating
Or_Y_std_noneating
Gy_Y_std_noneating
Gy_Y_min_noneating
Ax_Y_max_noneating
Or_X_rms_noneating


column = ['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Acceleromet

df = pd.read_csv("IMUeatingfile.csv")
a = df['Number'].unique()
a.sort()

li = {}
for i in a:
  li[i] = df.loc[df['Number'] == i]

x1, y1 = {}, {}
# Mean of EMG1 for all Eating
means_eating = {}
for j in column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    means_eating[i] = df[col].max() # means_eating[1] is mean of EMG1
    #print("Done for person ", i)
  lists = sorted(means_eating.items()) # sorted by key, return a list of tuples
  u, v = zip(*lists) # unpack a list of pairs into two tuples
  x1[j], y1[j] = u, v
  #print("Done for col ",col )


#Ax_X_mean_eating = y1['Accelerometer X']
#Ax_X_std_eating = y1['Accelerometer X']
#Or_Y_std_eating = y1['Orientation Y']
#Gy_Y_std_eating = y1['Gyroscope Y']
#Gy_Y_min_eating= y1['Gyroscope Y']
#Ax_Y_max_eating = y1['Accelerometer Y']
#Or_X_rms_noneating = y1['Orientation X']



# Creating feature matrix
df_ = pd.concat([pd.DataFrame(list(Ax_X_mean_eating), columns = ['Ax_X_mean']),
pd.DataFrame(list(Ax_X_std_eating), columns = ['Ax_X_std']),
pd.DataFrame(list(Or_Y_std_eating), columns = ['Or_Y_std']),
pd.DataFrame(list(Gy_Y_std_eating), columns = ['Gy_Y_std']),
pd.DataFrame(list(Gy_Y_min_eating), columns = ['Gy_Y_min']),
pd.DataFrame(list(Ax_Y_max_eating), columns = ['Ax_X_max']),
pd.DataFrame(list(Or_X_rms_noneating), columns = ['Or_X_rms'])], axis=1)


df_
```

	Ax_X_mean	Ax_X_std	Or_Y_std	Gy_Y_std	Gy_Y_min	Ax_X_max	Or_X_rms
0	0.946425	0.063256	0.441103	17.903207	-66.875	0.188	0.571991
1	0.799442	0.090656	0.069035	12.515566	-57.312	0.122	0.823019
2	0.831530	0.132440	0.090091	20.794830	-115.190	0.096	0.466557
3	0.907349	0.110793	0.076859	15.878719	-79.312	0.116	0.791635
4	0.793477	0.137349	0.081490	19.122790	-162.630	0.484	0.781090
5	-0.922377	0.104120	0.103747	21.204174	-167.190	0.896	0.519368
6	0.817414	0.106410	0.074257	16.075812	-158.750	0.171	0.129079
7	0.706866	0.164320	0.258655	18.508217	-80.750	0.213	0.573601
8	0.824171	0.078671	0.044349	16.457886	-95.188	0.238	0.853004
9	0.900094	0.072330	0.079732	11.424031	-48.438	0.239	0.534476
10	0.961001	0.072090	0.092511	14.912092	-107.810	0.336	0.496079
11	0.865662	0.082335	0.100274	17.200280	-60.875	0.040	0.434493
12	0.765236	0.111050	0.047656	15.368677	-106.440	0.220	0.832612
13	0.921648	0.072889	0.086133	13.603156	-51.312	0.328	0.275756
14	0.810924	0.204061	0.088666	19.299456	-102.000	0.189	0.159239
15	0.917939	0.072872	0.130071	19.439917	-109.690	0.750	0.255497
16	0.822400	0.125896	0.127769	20.743668	-161.750	0.088	0.150776
17	0.855566	0.173436	0.095532	20.533187	-50.500	0.172	0.516923
18	0.655015	0.134251	0.100463	17.377231	-118.380	0.257	0.888755
19	0.836372	0.096878	0.077936	16.759503	-68.500	0.277	0.829327
20	0.919597	0.092945	0.076088	11.228494	-139.000	0.520	0.612252
21	0.945357	0.070709	0.080690	16.294569	-71.188	0.083	0.611393
22	0.904116	0.099822	0.086797	12.480392	-84.375	0.259	0.720507
23	0.839873	0.152271	0.094322	21.656529	-99.188	0.374	0.154904
24	0.796980	0.156654	0.118879	18.021987	-85.500	0.110	0.852683
25	0.819607	0.078447	0.066158	14.075933	-84.812	0.167	0.726616
26	0.686537	0.138646	0.055093	16.502030	-66.625	0.801	0.708978
27	0.872800	0.147529	0.085286	14.255715	-72.812	0.174	0.725638
28	0.804260	0.172348	0.099691	27.634483	-96.438	0.172	0.851203

```
import numpy as np
```

```
from sklearn.decomposition import PCA


my_model = PCA(n_components=7)
my_model.fit_transform(df_)

my_model.explained_variance_
```

```
array([1.19191826e+03, 1.30266662e+01, 1.03436727e-01, 5.33072083e-02,
       2.63282838e-02, 5.04435897e-03, 9.12386831e-04])
```

```
my_model.explained_variance_ratio_
```

```
array([9.89033837e-01, 1.08093098e-02, 8.58300659e-05, 4.42334299e-05,
       2.18467696e-05, 4.18572471e-06, 7.57083334e-07])
```

```
my_model.explained_variance_ratio_.cumsum()
```

```
array([0.98903384, 0.99984315, 0.99992898, 0.99997321, 0.99999506,
       0.99999924, 1.        ])
```

```
my_model.components_
```

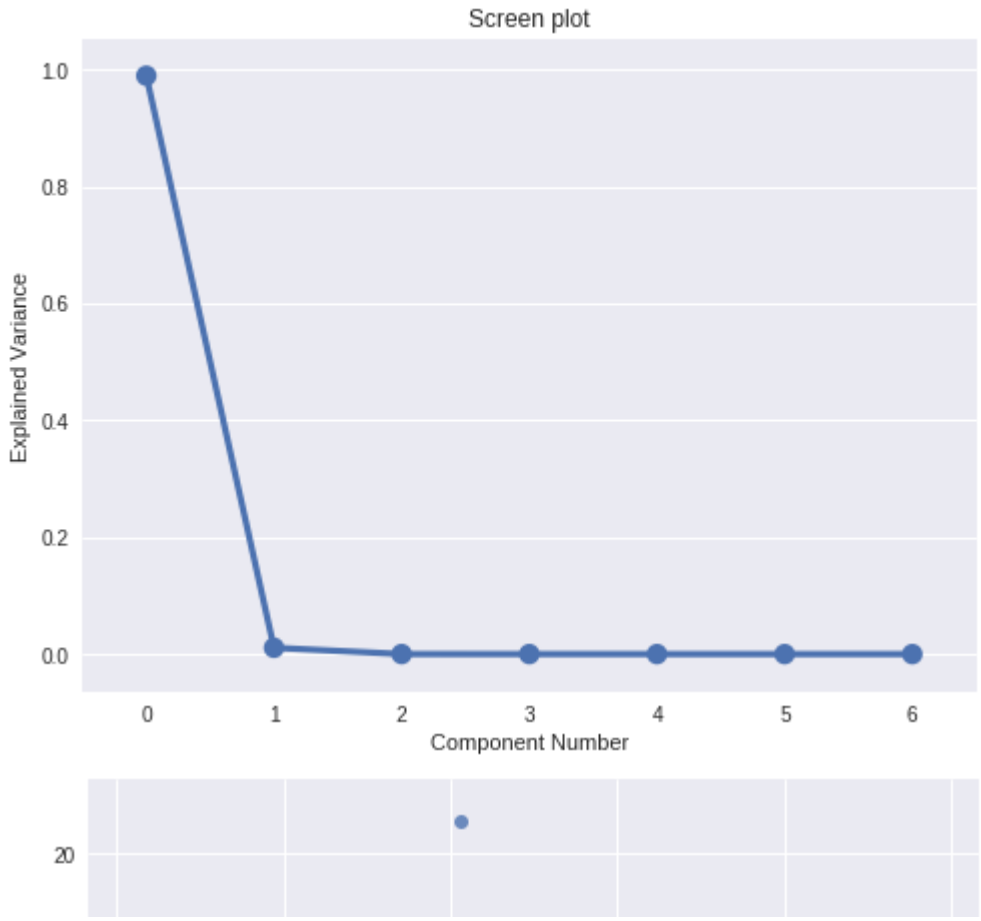
```
array([[-4.02068637e-03,  1.40216831e-04, -2.33232275e-04,
         3.15710050e-02, -9.99489843e-01,  2.07397317e-03,
        -1.66870588e-03],
       [-1.11352833e-02,  5.48550830e-03,  4.25771533e-03,
         9.99401218e-01,  3.16171166e-02, -1.71646468e-03,
        -4.73319173e-03],
       [-8.94878958e-01, -8.00944384e-03, -9.17406131e-03,
        -8.46998328e-03,  3.88790068e-03,  4.09508717e-01,
         1.76802408e-01],
       [-9.43583894e-02, -1.18286177e-02,  4.40540581e-02,
        -5.47825708e-03,  2.25965767e-03,  2.13746793e-01,
        -9.71232727e-01],
       [ 4.35873289e-01, -2.03812704e-02, -3.83427027e-02,
         7.36715487e-03,  7.18090078e-05,  8.86132907e-01,
         1.51139367e-01],
       [ 1.25774117e-02, -1.10629528e-01,  9.92149808e-01,
        -3.18149907e-03, -4.29845996e-04,  2.55657487e-02,
         5.07716519e-02],
       [ 2.06647892e-03,  9.93534868e-01,  1.10115801e-01,
        -5.85898541e-03, -2.16521933e-05,  2.68800150e-02,
        -1.35757441e-03]])
```

```
from sklearn.decomposition import PCA
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

n_components = 7

# Do the PCA.
pca = PCA(n_components=n_components)
reduced = pca.fit_transform(df_)
```

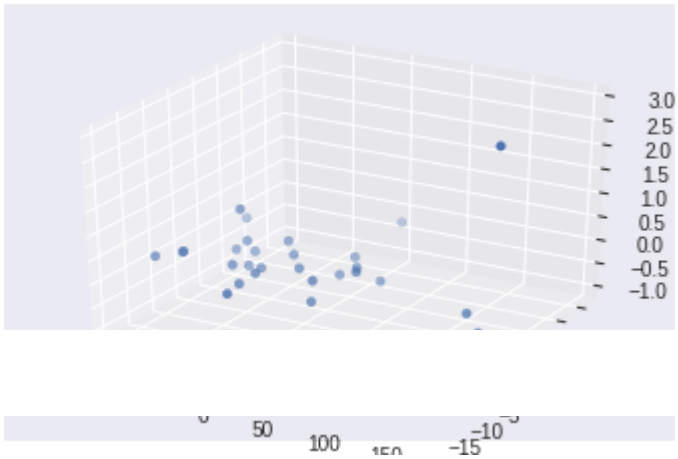
```
# Append the principle components for each entry to the dataframe
for i in range(0, n_components):
    df_['PC' + str(i + 1)] = reduced[:, i]

display(df_.head())

# Do a screen plot
ind = np.arange(0, n_components)
(fig, ax) = plt.subplots(figsize=(8, 6))
sns.pointplot(x=ind, y=pca.explained_variance_ratio_)
ax.set_title('Screen plot')
ax.set_xticks(ind)
ax.set_xticklabels(ind)
ax.set_xlabel('Component Number')
ax.set_ylabel('Explained Variance')
plt.show()

# Show the points in terms of the first two PCs
g = sns.lmplot('PC1',
               'PC2',
               data=df_,
               fit_reg=False,
               scatter=True,
               size=7)

plt.show()

# Plot a variable factor map for the first two dimensions.
(fig, ax) = plt.subplots(figsize=(12, 12))
for i in range(0, len(pca.components_)):
    ax.arrow(0,
             0,  # Start the arrow at the origin
             pca.components_[0, i],  #0 for PC1
             pca.components_[1, i],  #1 for PC2
             head_width=0.1,
             head_length=0.1)

    plt.text(pca.components_[0, i] + 0.05,
         pca.components_[1, i] + 0.05,
         df_.columns.values[i])

an = np.linspace(0, 2 * np.pi, 100)
plt.plot(np.cos(an), np.sin(an))  # Add a unit circle for scale
plt.axis('equal')
ax.set_title('Variable factor map')
plt.show()
```

	Ax_X_mean	Ax_X_std	Or_Y_std	Gy_Y_std	Gy_Y_min	Ax_X_max	Or_X_rms	PC
0	0.946425	0.063256	0.441103	17.903207	-66.875	0.188	0.571991	-56.93102
1	0.799442	0.090656	0.069035	12.515566	-57.312	0.122	0.823019	-76.38719
2	0.831530	0.132440	0.090091	20.794830	-115.190	0.096	0.466557	39.83334
3	0.907349	0.110793	0.076859	15.878719	-79.312	0.116	0.791635	-32.19807
4	0.793477	0.137349	0.081490	19.122790	-162.630	0.484	0.781090	134.56023

```
/usr/local/lib/python3.6/dist-packages/seaborn/categorical.py:1428: FutureWarning: r
  stat_data = remove_na(group_data)
```




```
%matplotlib inline
import numpy as np
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.axes(projection='3d')


# Data for three-dimensional scattered points
zdata = df_['PC3']
xdata = df_['PC1']
ydata = df_['PC2']
ax.scatter3D(xdata, ydata, zdata);
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
50
100
−15
−10