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

```
df2 = pd.read_csv("IMUnoneatingfile.csv", header = None)
```

```
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWar
  interactivity=interactivity, compiler=compiler, result=result)
```

```
# Non-eating Data
df.head()
```

	Number	Timestamp	Orientation X	Orientation Y	Orientation Z	Orientation W	Acce
354234	41	1.500000e+12	0.643	0.057	-0.177	-0.743	
354235	41	1.500000e+12	0.644	0.057	-0.178	-0.742	
354236	41	1.500000e+12	0.644	0.055	-0.179	-0.741	
354237	41	1.500000e+12	0.645	0.054	-0.179	-0.741	
354238	41	1.500000e+12	0.646	0.053	-0.180	-0.740	

```
# 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
Ax_Y_fft_noneating
Or_X_rms_noneating
```

```
# Creating feature matrix
df_ = pd.concat([pd.DataFrame(list(Ax_X_mean_noneating), columns = ['Ax_X_mean']),
pd.DataFrame(list(Ax_X_std_noneating), columns = ['Ax_X_std']),
pd.DataFrame(list(Or_Y_std_noneating), columns = ['Or_Y_std']),
pd.DataFrame(list(Gy_Y_std_noneating), columns = ['Gy_Y_std']),
pd.DataFrame(list(Gy_Y_min_noneating), columns = ['Gy_Y_min']),
pd.DataFrame(list(Ax_Y_max_noneating), columns = ['Ax_X_max']),
pd.DataFrame(list(Or_X_rms_noneating), columns = ['Or_X_rms'])], axis=1)
```

```
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([5.19513215e+03, 1.38985081e+01, 2.19950232e-01, 9.75090316e-02,
       5.18399471e-02, 5.53256248e-03, 4.37693249e-04])
```

```
my_model.explained_variance_ratio_
```

```
array([9.97259999e-01, 2.66796412e-03, 4.22217495e-05, 1.87178793e-05,
       9.95122050e-06, 1.06203328e-06, 8.40198009e-08])
```

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

```
array([0.99726   , 0.99992796, 0.99997018, 0.9999889 , 0.99999885,
       0.99999992, 1.        ])
```

```
pca.components_
```

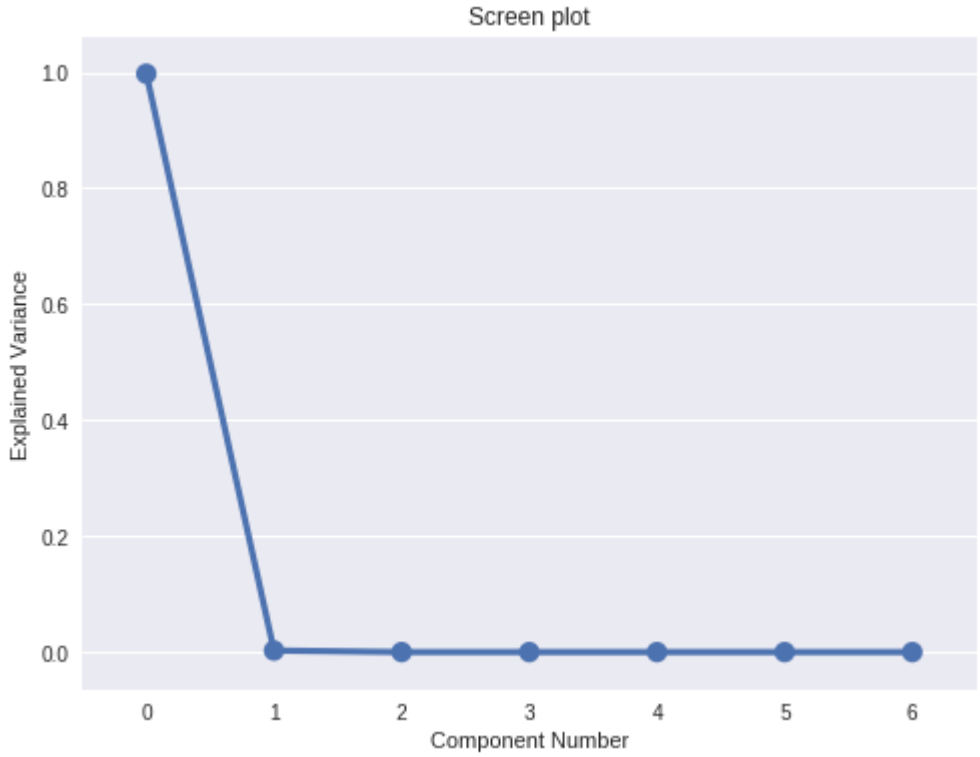
```
array([[-3.38876641e-05,  2.24545565e-05, -2.45783003e-05,
         4.95240162e-03, -3.01470339e-01,  4.38014728e-04,
        -7.02209197e-05,  9.53462589e-01,  1.38777878e-17,
        -9.71445147e-17, -1.11022302e-16,  8.80914265e-20,
         6.09863722e-20,  1.64575855e-20],
       [-1.35905151e-03,  1.30473104e-03, -1.84040407e-04,
         3.01421789e-01,  4.95283042e-03, -1.53525363e-04,
        -5.08515996e-03, -8.04911693e-16,  9.53462589e-01,
         3.65506236e-15, -1.31838984e-16, -9.62229428e-18,
        -8.35312129e-18,  2.10587694e-18],
       [-9.07271619e-02,  4.34835508e-03, -3.21131090e-04,
         1.31610368e-04,  4.23048795e-04,  2.86420005e-01,
         2.49408491e-02,  4.46864767e-15, -2.33320308e-15,
         9.53462589e-01,  4.52762827e-16, -4.34451669e-17,
        -4.80933661e-17,  3.31317268e-17],
       [-3.01568065e-01, -1.46988394e-03,  4.13926865e-04,
        -1.53540302e-03, -1.27324622e-04, -9.47992698e-02,
        -8.06816294e-03, -2.06779038e-15, -4.38885039e-16,
         9.36750677e-17,  9.48683298e-01,  5.12878450e-18,
        -1.66804292e-16, -3.61949892e-17],
       [-1.39796218e-04, -2.06162528e-03,  1.45317522e-02,
        -5.27764333e-03,  2.50875409e-05,  2.74049009e-02,
        -3.14651673e-01,  9.08322896e-15, -8.84166872e-17,
         1.33817653e-16,  9.05308814e-17,  9.48683298e-01,
        -1.46648694e-16, -2.24367879e-17],
       [ 2.99458165e-04, -2.64686305e-03,  3.15881244e-01,
         4.50799613e-04, -2.30909267e-05, -7.76260970e-04,
         1.45305754e-02,  1.91448420e-14,  3.12933273e-15,
        -3.37024245e-16, -6.61363325e-16,  4.85147517e-17,
         9.48683298e-01, -3.12783251e-16],
       [-2.16622750e-05,  3.16170699e-01,  2.74681563e-03,
        -1.40840953e-03, -5.95271696e-06, -4.60096998e-03,
        -2.32181389e-03, -1.24678046e-13,  6.33521013e-15,
        -1.71598846e-14, -1.66047731e-14, -3.47188641e-16,
         2.17935225e-15,  9.48683298e-01]])
```

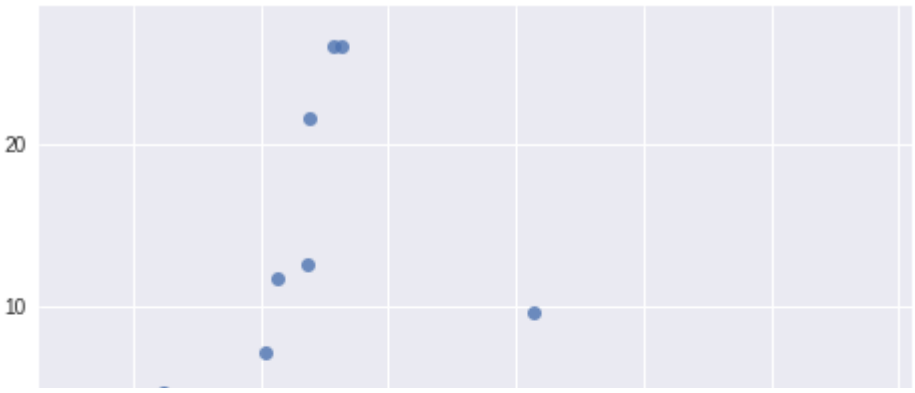
```
from sklearn.decomposition import PCA
import seaborn as sns
```

```
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	PC1
0	0.943284	0.061066	0.435325	15.615816	-105.310	0.443	0.598434	-121.112883
1	0.806906	0.080420	0.063509	14.365574	-201.810	0.718	0.825139	198.831079
2	0.810425	0.164748	0.100523	23.969699	-165.130	1.747	0.490698	77.722251
3	0.912240	0.091031	0.063271	18.884034	-229.810	0.226	0.787798	291.927715
4	0.825986	0.119239	0.077878	15.079134	-83.562	0.158	0.768975	-193.263617

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






```
df_.head()
```

	Ax_X_mean	Ax_X_std	Or_Y_std	Gy_Y_std	Gy_Y_min	Ax_X_max	Or_X_rms	PC1
0	0.943284	0.061066	0.435325	15.615816	-105.310	0.443	0.598434	-121.112883

```
%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, c="green");
```