Daniel Gil

Student ID: 905923

COMP90073-Security Analytics Assignment 1

First it is necessary to import some python packages. Sklearn and Pandas are default packages in python 3.5+

In [157]:

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

import seaborn as sns
sns.set(style="ticks", color_codes=True)
```

Load data exported from splunk

In [158]:

```
df_normal=pd.read_csv("feat2_attack.csv")
df_attack=pd.read_csv("feat2_normal.csv")
```

We can merge together both datasets just to explore a general relation between the variables

In [160]:

```
df_normal['attack']=0
df_attack['attack']=1
df=pd.concat([df_normal,df_attack])
print(df.columns)
```

```
Index(['start', 'finish', 'src_ip', 'dst_ip', 'duration', 'num_dst_p
ort',
       'total_bytes', 'num_packets', 'mbps', 'attack'],
      dtype='object')
```

Let's perform some basic exploration to look for correlation in the data

We are interested in some of these features to perform a cluster analysis

In [5]:

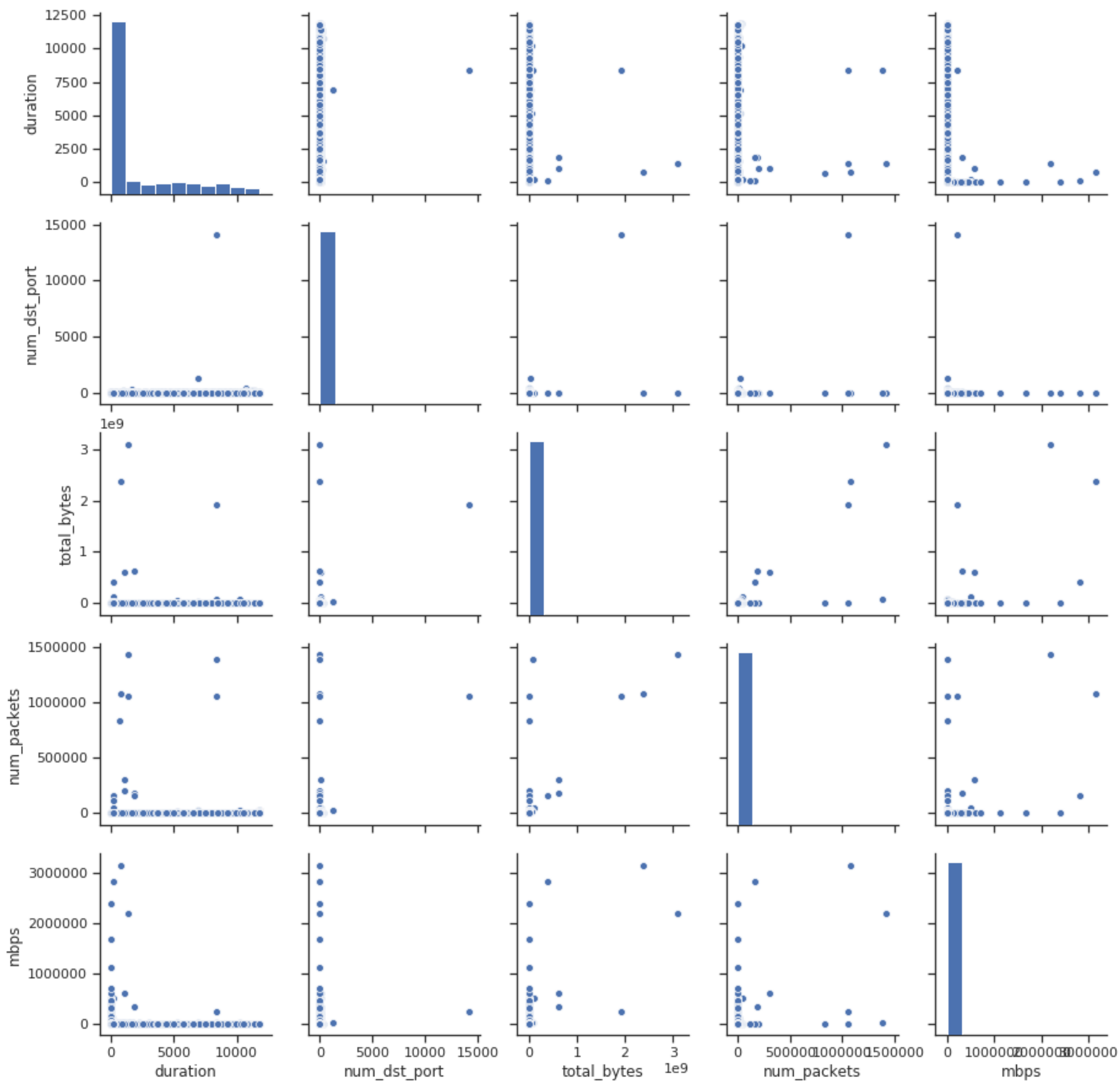
```
selected_features=[ 'duration', 'num_dst_port',
       'total_bytes', 'num_packets', 'mbps']
```

In [166]:

```
sns.pairplot(df_normal[selected_features])
```

Out[166]:

```
<seaborn.axisgrid.PairGrid at 0x7ffaaeca2d68>
```



As one can be interested to visualize the clusters, we can make some dimensionality reduction. It's important to scale variables as we have different kind of measures.

In [101]:

```
X_Normal = StandardScaler().fit_transform(df_normal[selected_features])
X_Attack = StandardScaler().fit_transform(df_attack[selected_features])
X_All= StandardScaler().fit_transform(df[selected_features])
```

Dimensionality reduction will be performed using PCA, we are interested to produce some features that explain most of the variability of the data

In [7]:

```
pca = PCA(n_components=3,random_state=42).fit(X_All)
print("Variability explained by the PC:",sum(pca.explained_variance_ratio_) )
reduced_data = pca.transform(X_All)
```

```
Variability explained by the PC: 0.8396734628953998
```

In [8]:

```
pd.DataFrame(pca.components_.T, index=selected_features,columns=["PC1","PC2","PC
3"])
```

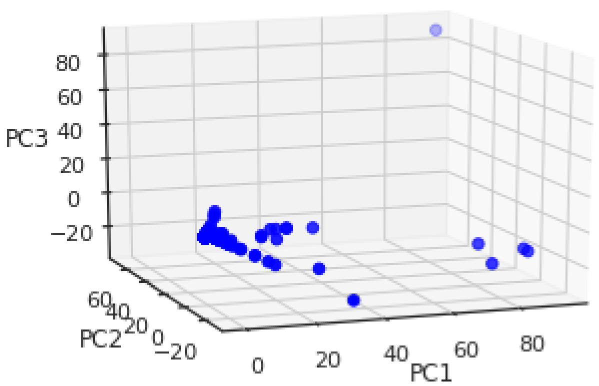
Out[8]:

	PC1	PC2	PC3
duration	-0.000654	0.772492	-0.633709
num_dst_port	0.297597	0.514425	0.656353
total_bytes	0.617046	-0.018323	-0.036169
num_packets	0.585354	0.033046	0.005810
mbps	0.433648	-0.370400	-0.407764

We can plot the data in 3 dimensions:

In [9]:

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

# plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(reduced_data[:,0], reduced_data[:,1], reduced_data[:,2], c='blue', s=
30)
ax.view_init(15, 250)
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
plt.show()
```



From this plot we can clearly some groups of data and a clear data point as outliers. In fact, we can set up a kmeans model to group identify possible outliers

1. Set up training data with normal day

In [28]:

```
pca_normal = PCA(n_components=3,random_state=42).fit(X_Normal)
reduced_data_normal = pca_normal.transform(X_Normal)
```

In [29]:

```
pca_attack = PCA(n_components=3,random_state=42).fit(X_Attack)
reduced_data_attack = pca_attack.transform(X_Attack)
```

1. Train the kmeans model

In [30]:

```
from sklearn.cluster import KMeans

model = KMeans(n_clusters=1).fit(reduced_data_normal)
```

1. Make prediction with our attack day data

In [31]:

```
labels_attack=model.predict(reduced_data_attack)
```

In [32]:

```
# calculate centroid and labels
centroids = model.cluster_centers_
```

In [33]:

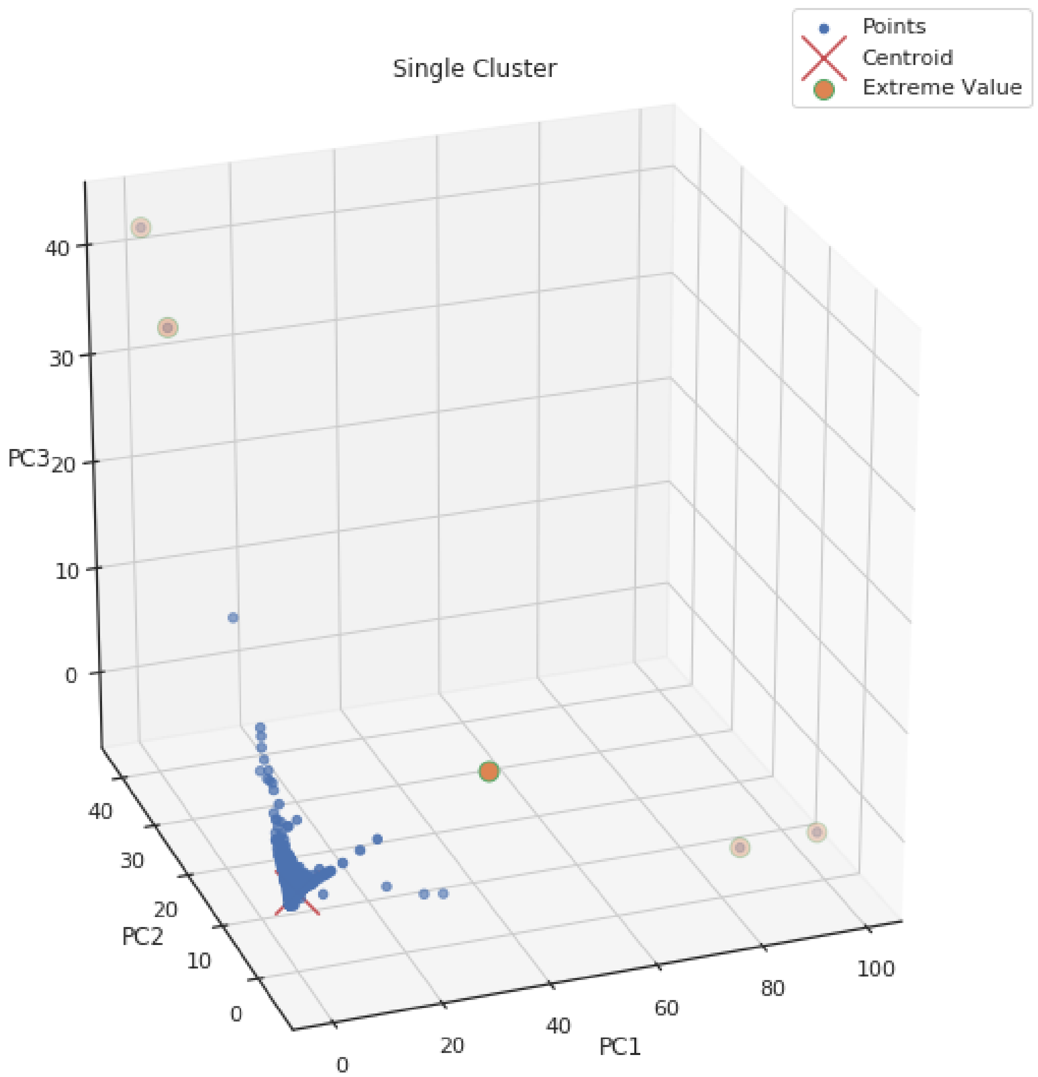
```
import numpy as np

# identify the 5 closest points
distances = model.transform(reduced_data_attack)

# argsort returns an array of indexes which will sort the array
# in ascending order. Reverse it with [::-1]
sorted_idx = np.argsort(distances.ravel())[::-1][:5]
```

In [55]:

```
fig = plt.figure(1, figsize=(8, 8))
plt.clf()
ax = Axes3D(fig, elev=4, azim=200)
#f, ax = plt.subplots(figsize=(7,5))
ax.view_init(25, 250)
ax.set_title('Single Cluster')
ax.scatter(reduced_data_attack[:, 0], reduced_data_attack[:, 1],reduced_data_att
ack[:, 2], label='Points')
ax.scatter(model.cluster_centers_[:, 0],
           model.cluster_centers_[:, 1],
           model.cluster_centers_[:, 2],
           label='Centroid', marker='x',s=500,color='r')
ax.scatter(reduced_data_attack[sorted_idx][:,0],reduced_data_attack[sorted_idx]
[:,1],reduced_data_attack[sorted_idx][:,2],label='Extreme Value', edgecolors='g'
, s=100)
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.legend(loc='best')
```

Out[55]:

```
<matplotlib.legend.Legend at 0x7ffab7949e10>
```



Now, recall principal components eigen vectors:

In [57]:

```
pd.DataFrame(pca.components_.T, index=selected_features,columns=["PC1","PC2","PC3"])
```

Out[57]:

	PC1	PC2	PC3
duration	-0.000654	0.772492	-0.633709
num_dst_port	0.297597	0.514425	0.656353
total_bytes	0.617046	-0.018323	-0.036169
num_packets	0.585354	0.033046	0.005810
mbps	0.433648	-0.370400	-0.407764

Extreme values are identified for higher values in PC1 and higher values for PC3. This is, if we have traffic flowing between 2 hosts with a significant payload, requests and concentrated in small amount of time, and in adition we have an abnormal number of ports used, then this traffic is being classified as anomaly.

In [ ]:

```

```