Final Project

August 12, 2020

1 Final Project Instructions

Once completed, you will submit the project as jupyter notebooks (as a single .ipynb file) through e-mail at 'tahiya.chowdhury@rutgers.edu' for grading.

In the e-mail, use subject line: `CPSML: Final Project _ your full name`

Both the jupyter notebook and e-mail should contain your name in English.

1.1 Goals

In this project, you will perform: * Data analysis and Visualization on multiple sensor dataset * K-means Clustering on sensor data * Dimensionality Reduction and its effect on clustering

1.2 Data

You will use acclerometer data from sensor mounted on chest to perform the analysis. The data can be found here: https://sensor.informatik.uni-mannheim.de/#dataset_realworld_subject1

We will use Subject 1 data only. For subject 1, we will use the following activity data: * climbing down * lying * running

We will use data collected from Accelerometer sensor mounted on chest only.

To download the data: * go to the link: https://sensor.informatik.uni-mannheim.de/#dataset_realworld_subject1 * There are 8 activities data in that page. Go to `Running`. Beside the word `Accelerometer`. * Click on `csv`. This will download a zip file in your computer. Unzip the folder. Inside the folder, choose the `csv` file named `acc_running_chest.csv` and put it in your project notebook directory (where your Final project notebook is located.) * Do the same for Activity `Climbing Down` and `Lying`. Your notebook directory should now have 3 csv (acc_running_chest.csv, acc_climbingdown_chest.csv, acc_lying_chest.csv).

1.3 Libraries

For this project, you will be using `pandas`, `numpy`, `scikit-learn`, `seaborn`, and `matplotlib` library to analyze and experiment on an human acivity data collected using accelerometer sensor.

[1]:

```
# importing libraries
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
```

1.4 Data Analysis/Cleaning

- Read the 3 csv files using `pd.read_csv` into 3 dataframes.

[2]:

```
df_climb_down = pd.read_csv('./acc_climbingdown_chest.csv')  ## your data and␣
 ↪notebook file should be in the same folder for this to work
#df_climb_down
```

[4]:

```
## Read running data into df_running the same way



#df_running = ## write your code here
```

[5]:

```
## Read lying data into df_running the same way



#df_lying = ## write your code here
```

The first 10 lines of climbing down data should look like this:

[6]:

```
df_climb_down.head(10)
```

[6]:

```
   id      attr_time    attr_x    attr_y    attr_z
0   1  1435996968010  5.616797  8.064270  0.878073
1   2  1435996968032  5.589264  8.054693  0.869095
2   3  1435996968052  5.580884  8.060080  0.908001
3   4  1435996968073  5.588067  8.033744  0.884658
4   5  1435996968093  5.583877  8.060678  0.875679
5   6  1435996968113  5.596446  8.047510  0.858920
6   7  1435996968131  5.609016  8.051102  0.862511
7   8  1435996968150  5.600637  8.058883  0.869694
8   9  1435996968170  5.589264  8.024167  0.891242
9  10  1435996968230  5.547365  8.076241  0.959476
```

There are 5 columns in each dataframe: `id, attr_time, attr_x, attr_y, attr_z`. Remove the first 2 columns (id and time) and convert the dataframe into a numpy array. You can do so in following way:

[7]:

```
df_climb = df_climb_down[['attr_x', 'attr_y', 'attr_z']].values
print(df_climb.shape)           ## Check the shape of the array
print(type(df_climb))           ## Check type
```

```
(25435, 3)
<class 'numpy.ndarray'>
```

[8]:

```
# do the same for df_running


#df_run = # write you code here
```

[9]:

```
## do the same for df_lie


#df_lie = # write your code here
```

We will use 25000 observations from each type of activity to create a balanced dataset.

To do that, select the first 25000 rows (each row represent an observation) from all 3 activity types.

[10]:

```
df_climb = df_climb[:25000, :]

#df_run = ## write your code here
#df_lie =  ## write your code here
```

We will concatanate all 3 types into a single dataset. You can use `concatenate` function from numpy library.

[12]:

```
#df_full = np.concatenate(## write you code here)
```

`df_full` should contain 75000 rows and 3 columns.

1.5 Visualization

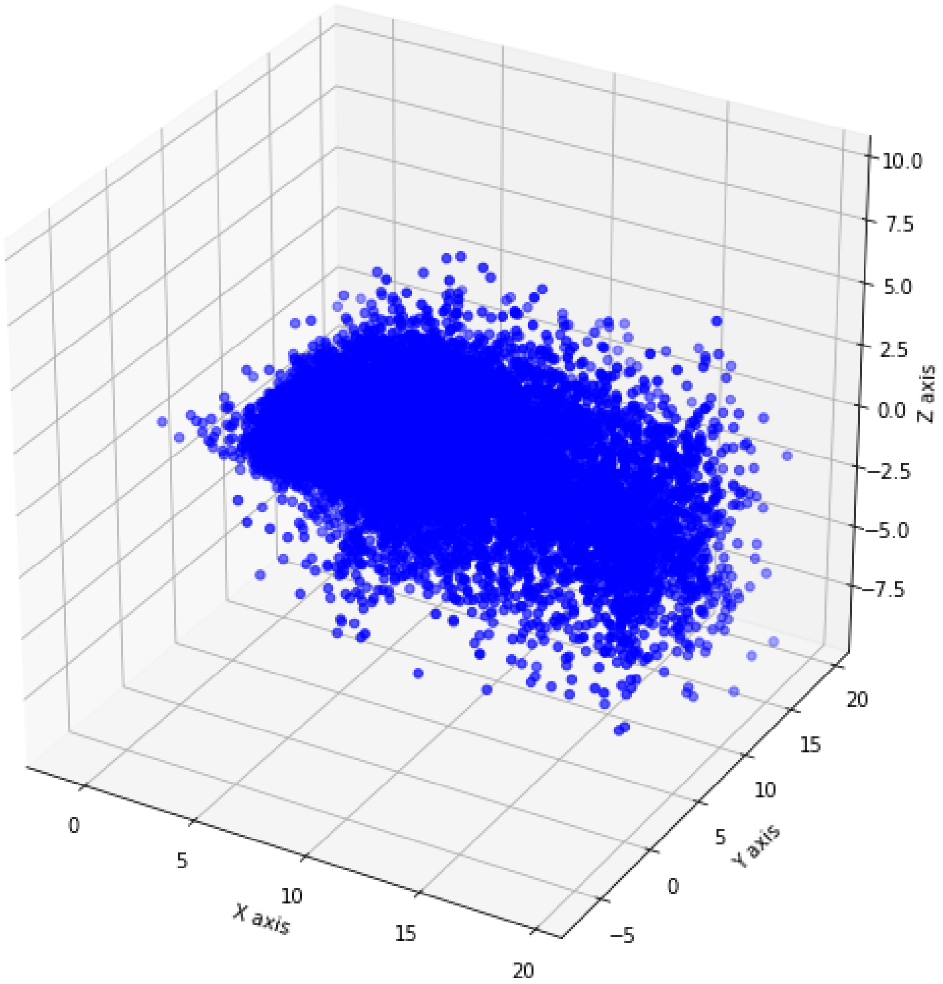
We will first plot the 3 dimensional data using matplotlib. For climbing down data it looks as following:

[14]:

```
from mpl_toolkits.mplot3d import Axes3D     ## this will allow us to perform␣
 ↪3-D plotting

fig = plt.figure(figsize = (10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df_climb[:, 0], df_climb[:, 1], df_climb[:, 2], marker = 'o', color␣
 ↪= 'b')

ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')
plt.show()
```



[15]:
```
# perform same 3d plotting for running data

# write your code here
```

[16]:
```
# perform same 3d plotting for lying data

# write your code here
```

1.5.1 Do the 3 activity data looks the same or different in the plotted figures? Observe the figures to understand why they are different.

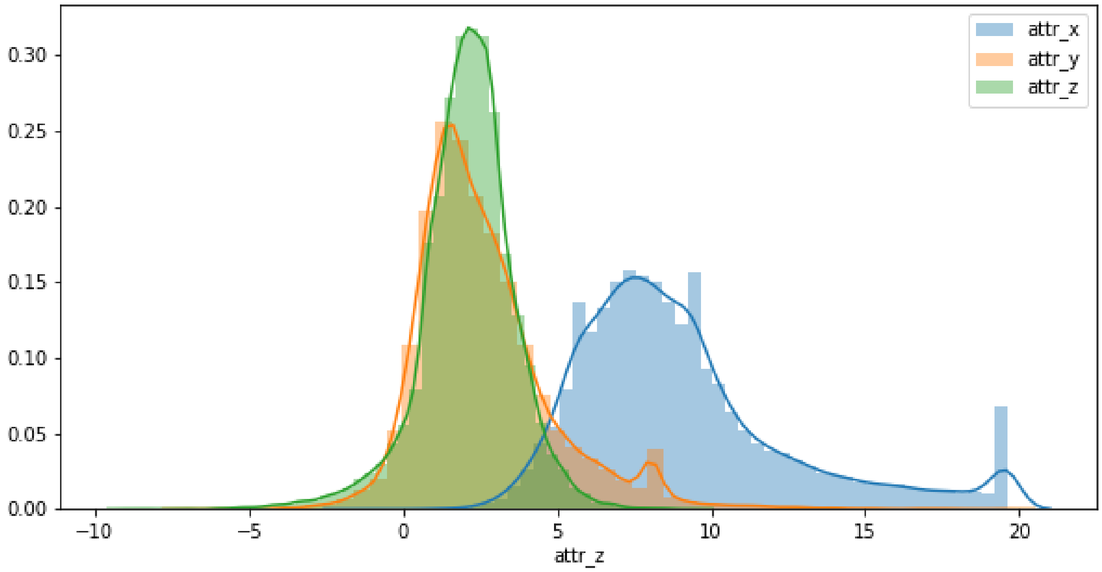
We can also plot them in their distribution.

```
[17]: df_climb_pair = df_climb_down[['attr_x', 'attr_y', 'attr_z']]

      #df_run_pair = df_running[['attr_x', 'attr_y', 'attr_z']]
      #df_lie_pair = df_lying[['attr_x', 'attr_y', 'attr_z']]
```

```
[18]: fig = plt.figure(figsize = (10, 5))
      ax = fig.add_subplot(111)
      ax = sns.distplot(df_climb_down['attr_x'], label = "attr_x")
      ax = sns.distplot(df_climb_down['attr_y'], label = "attr_y")
      ax = sns.distplot(df_climb_down['attr_z'], label = "attr_z")
      plt.legend()
```

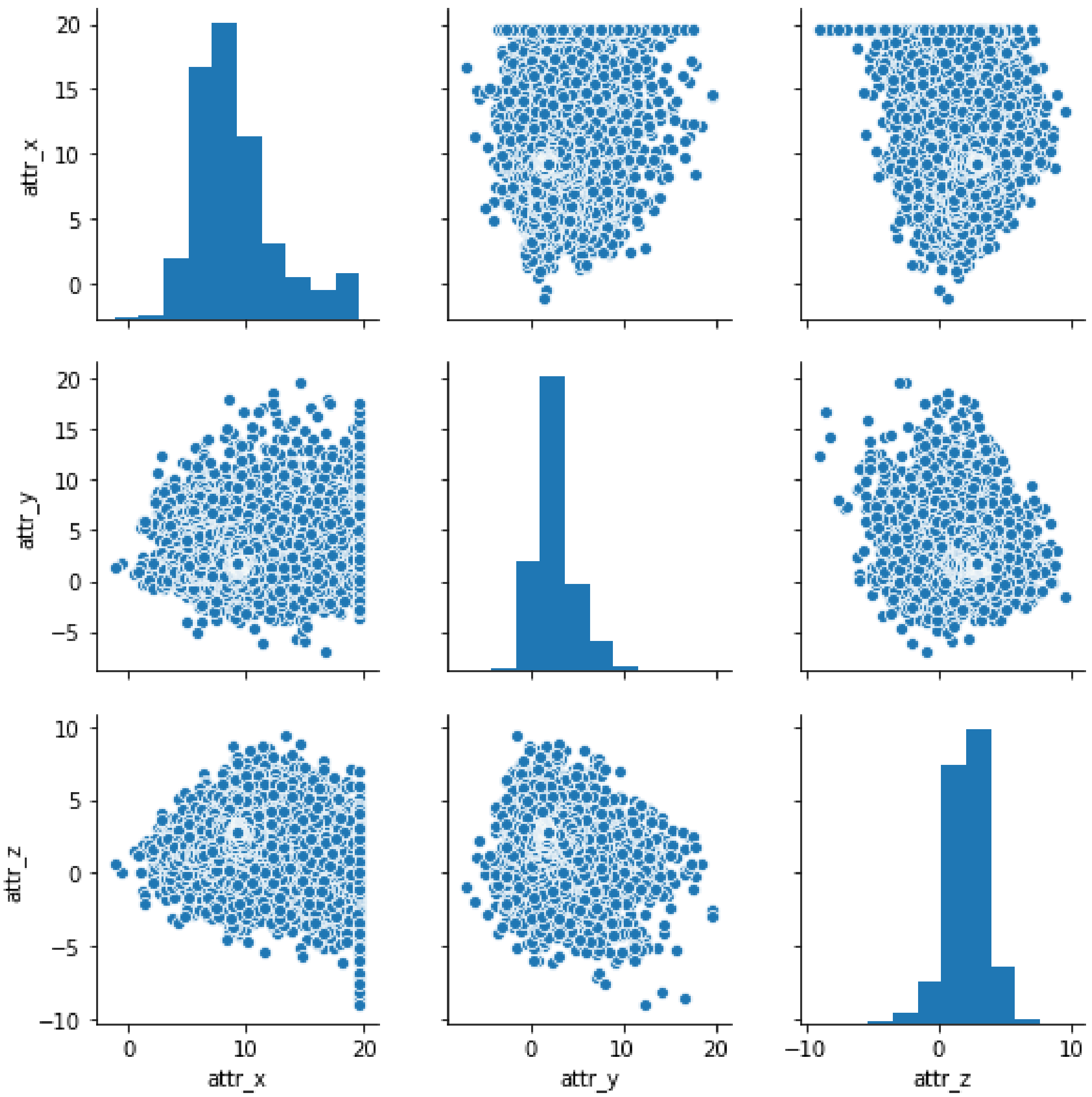
```
[18]: <matplotlib.legend.Legend at 0x11f436da0>
```



Plot similar distribution plots for running and lying dataset. * Are the distributions along different axis diffrent? * Are the distributions for different activities diffrent?

We can also plot the data in 2d plane to understand the correlation between x, y, z axis using `sns.pairplot`.

```
[19]: pair = sns.pairplot(df_climb_pair)
```



Do the same for running and lying data to understand relationship between x, y, z axis.

1.6 We will now apply k-means clustering method to cluster the combined dataset including the 3 types of activity.

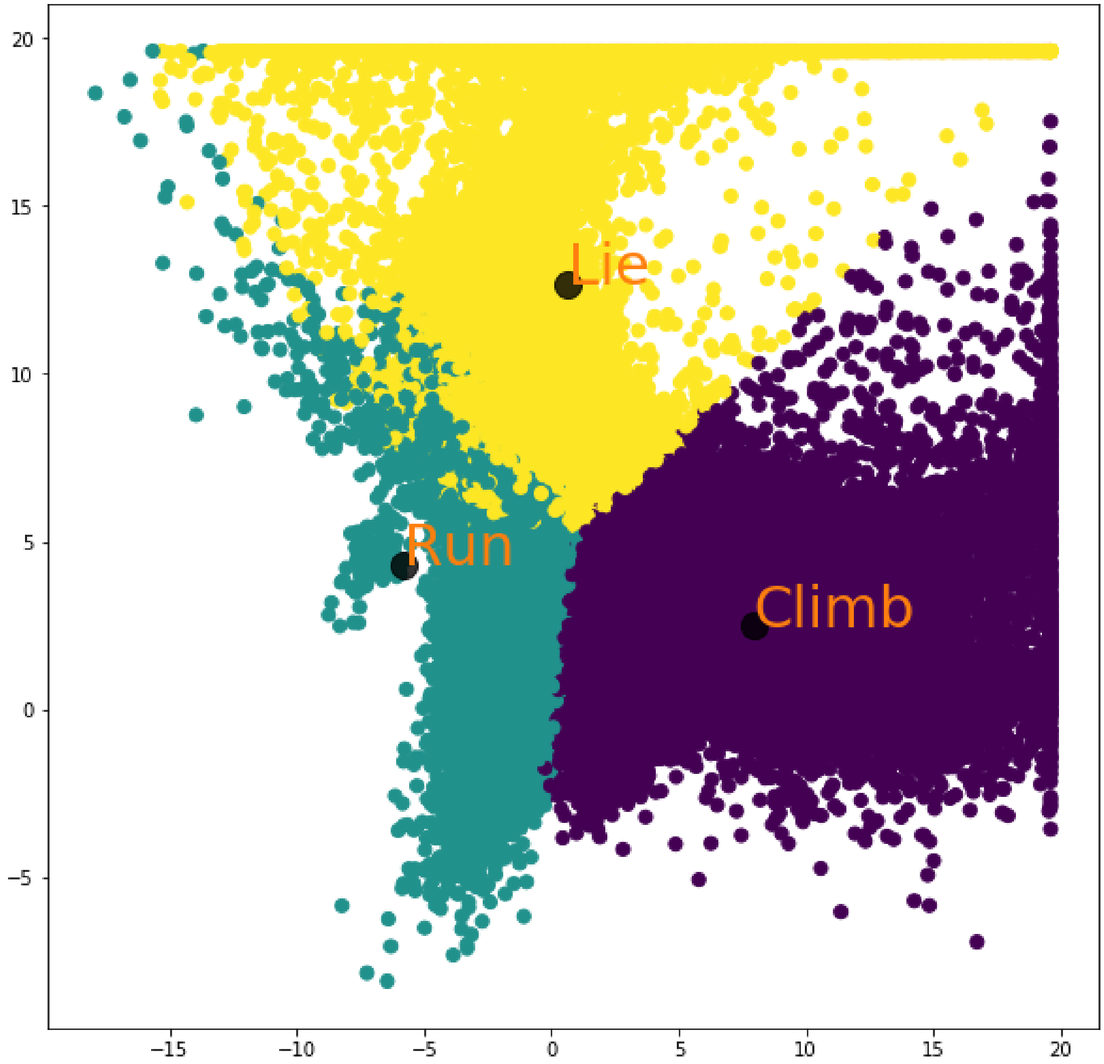
[20]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
kmeans.fit(df_full)
y_kmeans = kmeans.predict(df_full)

fig = plt.figure(figsize = (10, 10))
ax = fig.add_subplot(111)
```

```
ax.scatter(df_full[:, 0], df_full[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
ax.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.8);

labels = ['Climb', 'Run', 'Lie']
for i,txt in enumerate(labels):
    ax.annotate(txt, (centers[i, 0], centers[i, 1]), fontsize = 30, color =␣
 ↪'tab:orange')
```



[21]:
```
predicted_labels = kmeans.labels_
print(predicted_labels)
```

```
[0 0 0 … 1 1 1]
```

1.7 Dimensionality Reduction

We will apply dimensionality reduction to improve clustering result

[22]:

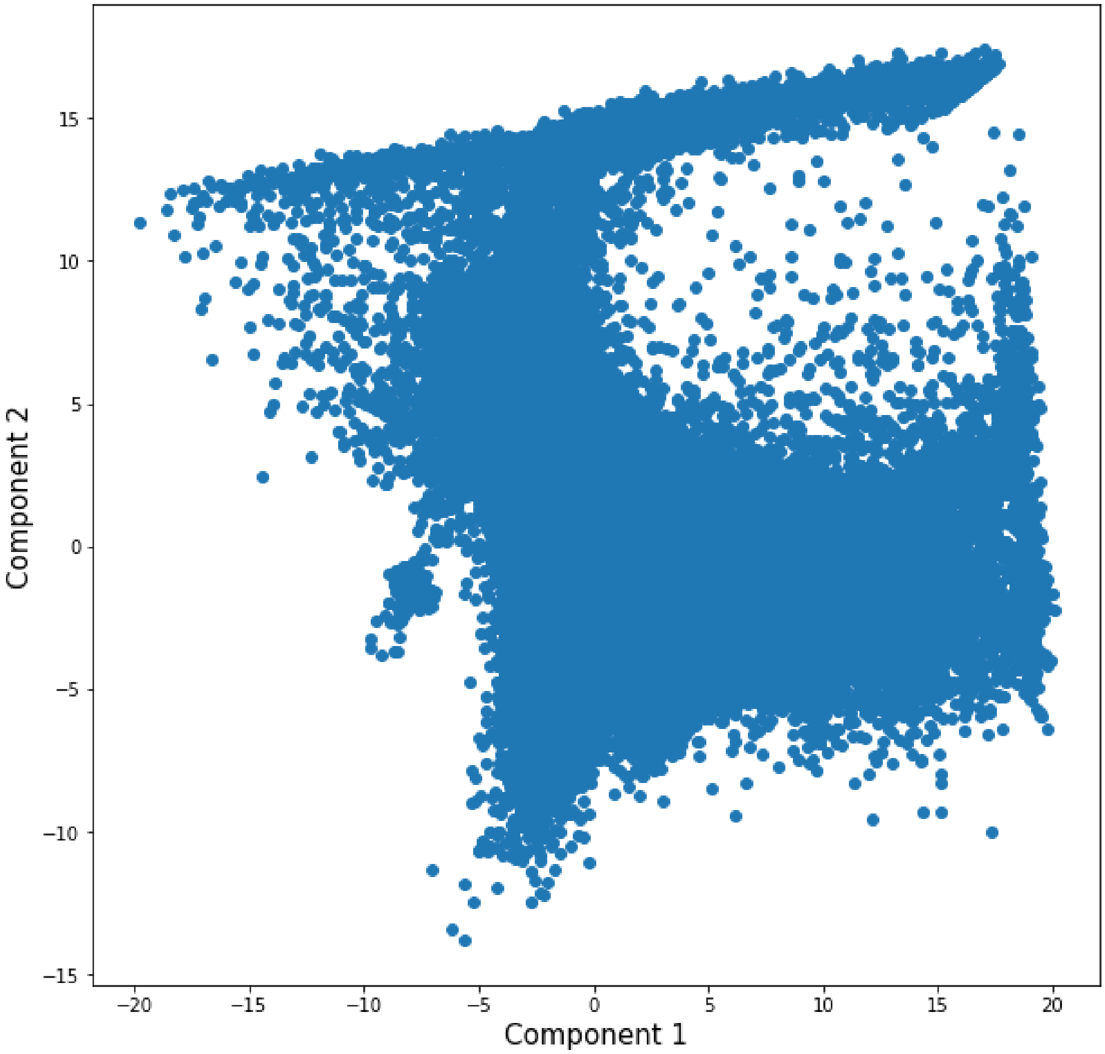
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)                 # 2 principal components
pca.fit(df_full)

fig = plt.figure(figsize = (10, 10))
ax = fig.add_subplot(111)
X_hat = pca.transform(df_full)
X_hat.shape
plt.scatter(X_hat[:,0], X_hat[:,1])
plt.xlabel("Component 1", fontsize = 15)
plt.ylabel("Component 2", fontsize = 15)
```

[22]:

```
Text(0, 0.5, 'Component 2')
```



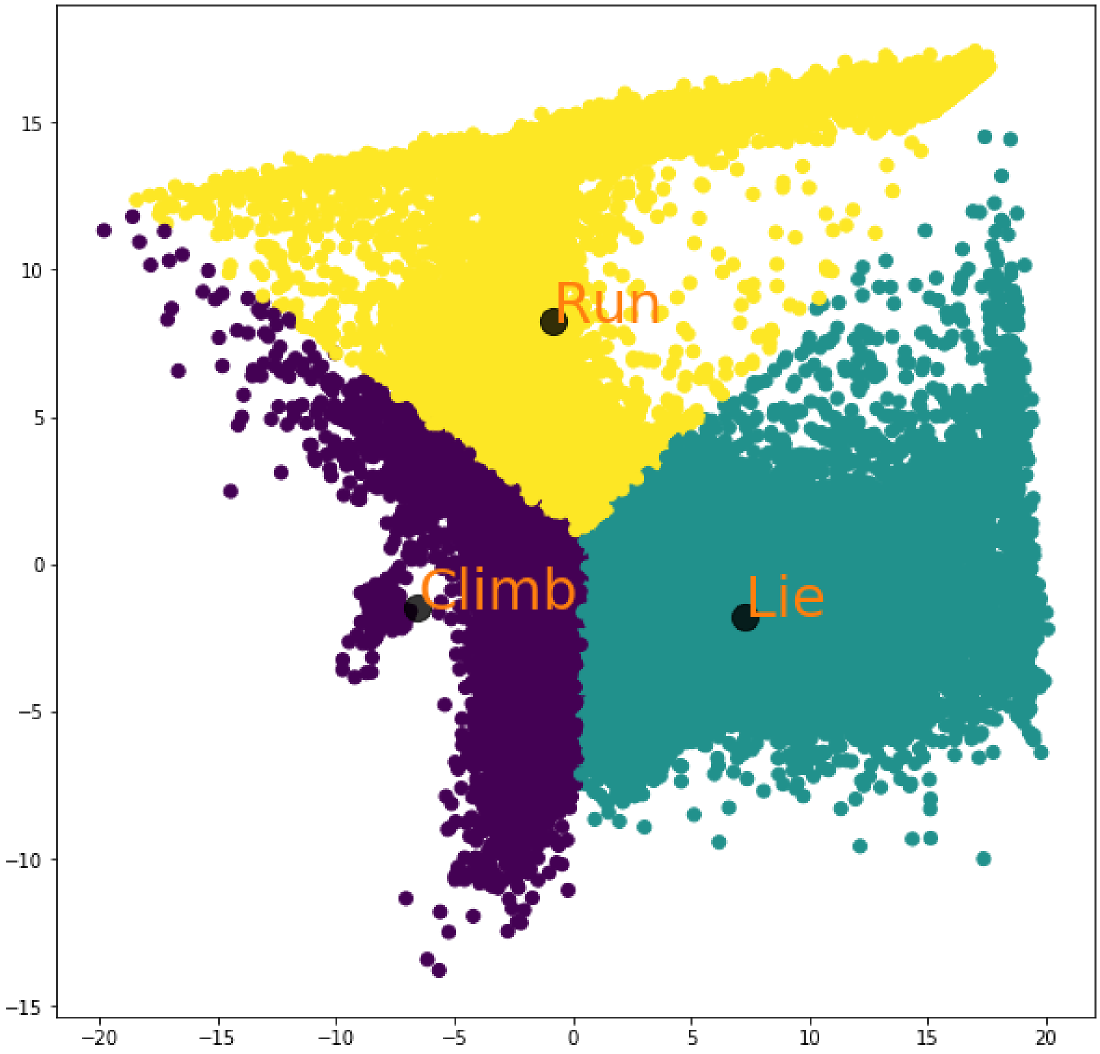
[23]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
kmeans.fit(X_hat)
y_kmeans = kmeans.predict(X_hat)

fig = plt.figure(figsize = (10, 10))
ax = fig.add_subplot(111)
ax.scatter(X_hat[:, 0], X_hat[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
ax.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.8);
#print(centers.shape)

labels = ['Climb', 'Lie', 'Run']
```

```python
for i,txt in enumerate(labels):
    ax.annotate(txt, (centers[i, 0], centers[i, 1]), fontsize = 30, color = 'tab:orange')
```



Are the clusters better than before? You can perform such analysis to any dataset and visualize the results.