CMPT 353: Computational Data Science

Simon Fraser University - Summer 2019

Final Report: Walking Data Classifier

Team Members:

Bowen Wang	301267523
Duo Lu	301368672
Peng Hang	301270540

Git Repo: https://github.com/MingXXI/Walking-Data-Classifier.git

Table of Content

Introduction

Computational data science is a field of constructing mathematical models, developing quantitative analysis techniques, and using computers to analyze and solve scientific real-life problems (Schimid College, 2019). In our project, we are going to build a model based on data from cellphone sensor to identify the state of motion (walking on flat, going up or down stairs, or fall down unexpectedly). Through the project we collect our own data, clean and transform the data, and pass the data into different machine learning model. Although we are able to classify sensor data fraction by using our model, we plan to collect more data and use more sensor concurrently to get more feature so that we can get accurate data with shorter time fraction. In short, we are going to train a machine learning model to classify a 3-second data into several motion states.

Approaches and Methods

Data Collection

- **Initial Idea**

  The first thing we need to figure out is what tool is best to use to collect the sensor data. According to Baker's advice, we use app *Physics Toolbox Suite* by *Vieyra Software* to collect sensor data.

  Let's focus on our question again. For a given 3-seconds sensor data (roughly 2000 points), we classify it to one of eight motion states: holding the phone while walking, walking with phone inside of the pocket, holding the phone while walking downstairs, walking downstairs with phone in the pocket, walking upstairs same as downstairs, and the most important category: fall down situation. Now, as there are several sensors built in the cellphone (G-force sensor, acceleration sensor, gyroscope sensor etc.), which can be used in our project comes to be our next question.

  (For convenient, walking-hold, walking-in-pocket, downstairs/upstairs-hold, downstairs/upstairs-in-pocket, fall-down-hold/in pocket representing all the categories)

- **Sensor Choosing**
  - **First Try**

  As the sensor app can record two sensors' data at the same time, we decide to use **G-force** and **light-meter** as first combination.

  G-force meter measures the ratio of normal force to gravitational force (FN/Fg). The G-force changes **whenever the mobile device accelerates speeds up, slow down, or**

change directions. That’s probably what we need. **Light-meter** is a sensor to classify whether phone is holding by us or in the pocket. While light-meter is low whenever the phone in pocket and high whenever we hold the phone.

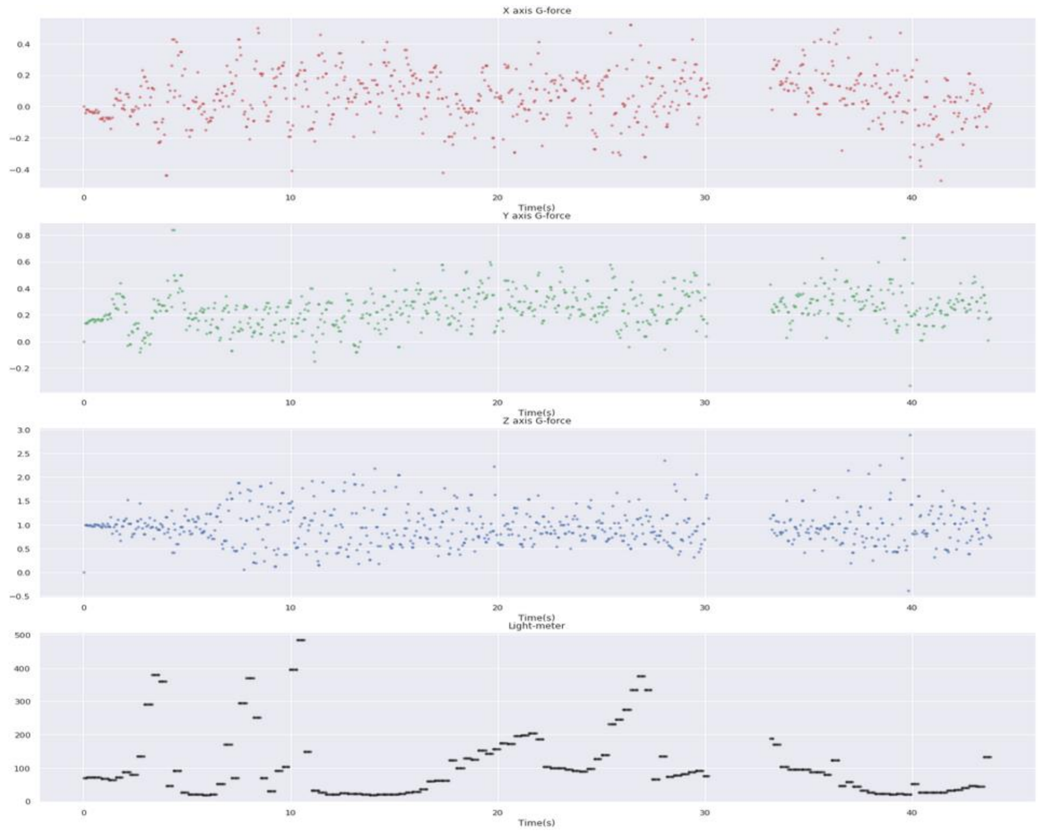


Table I: Data from G-Force and Light-Meter, plot the g-force and light-meter graph for walking-hold by hand sensor data, the first 3 subplots are for 3 dimensions g-force and the last one is light-meter. There is a gap occur in period from 30 to 35 second. For light-meter, inconsistent and also not much important information tells.

After taking a look at the plot of g-force and light-meter. We didn’t get much helpful information relates to our project even if we use some filter like Butterworth. We realize this is not a good combination of sensors.

■ Second Try

We tried to use three sensors like g-force, acceleration and light-meter. But we realize the app reduce its data sampling rate when multiple sensor active at same time. Thus, we step back to think about which two sensors are most significant for us rather than collecting useless datasets (Fortunately, dual sensor performs same as single sensor which at least better than multiply sensor!).

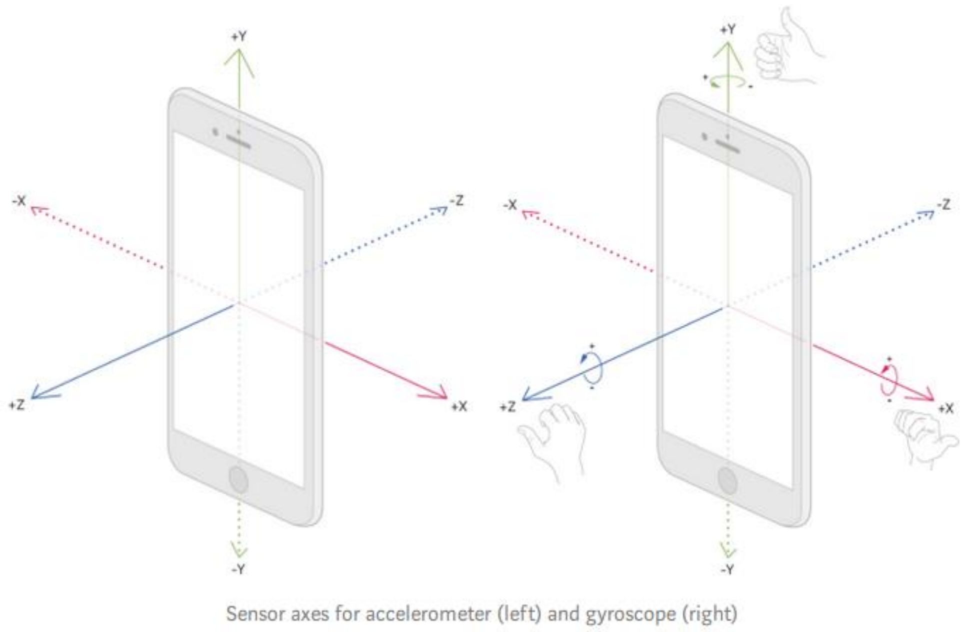


Table II: Sensor Axes (Malyi, 2017)

- **Final Decision for Sensor**

  We start to focus on the motion states rather than sensor. Compare to walking, the move of going downstairs/upstairs has greater range. Thus, we decided to use linear accelerometer as the first sensor. Because accelerometer provides changes in device's velocity along 3 axes which can tell us change in level of move's strenuousness. Moreover, when people fall down, the phone would have a significant rotate. So, gyroscope which can delivers the rate when a device rotates may help us. As a result, we decided to use **accelerometer** and **gyroscope** for our sensor. However, we finally discard the light-meter because it has bad performance and we have no idea how to deal with it. But after the further discussion, we realized that state of phone in pocket or hold may really different from each other and Machine Learning model may perform very well on it.

  Acceleration of downstairs held by hand sensor data in three second. There are 4 subplots in it. The first three is x,y,z axes acceleration and the last one is total acceleration. We can easily find some regularity from graph above. In addition, subplot of z axis might tell us something significant. Why? Most of the time, phone in our pocket is facing to our leg and z axis might move with the legs. And the maximum points are probably the points which our legs suspend on the stairs. Also, we can get more information from the total acceleration.

  Finally, we get our raw data and the structure is shown in Table V.

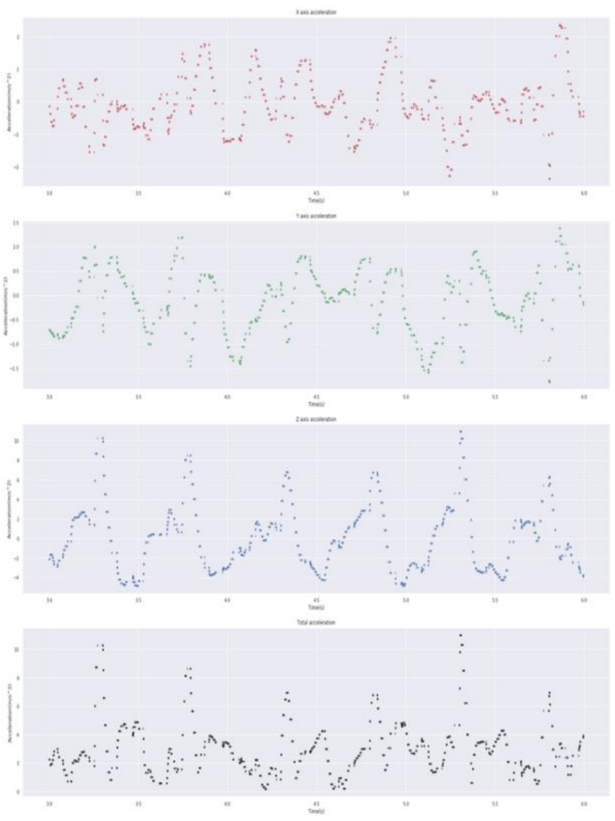


Table III: Accelerometer Data Sample

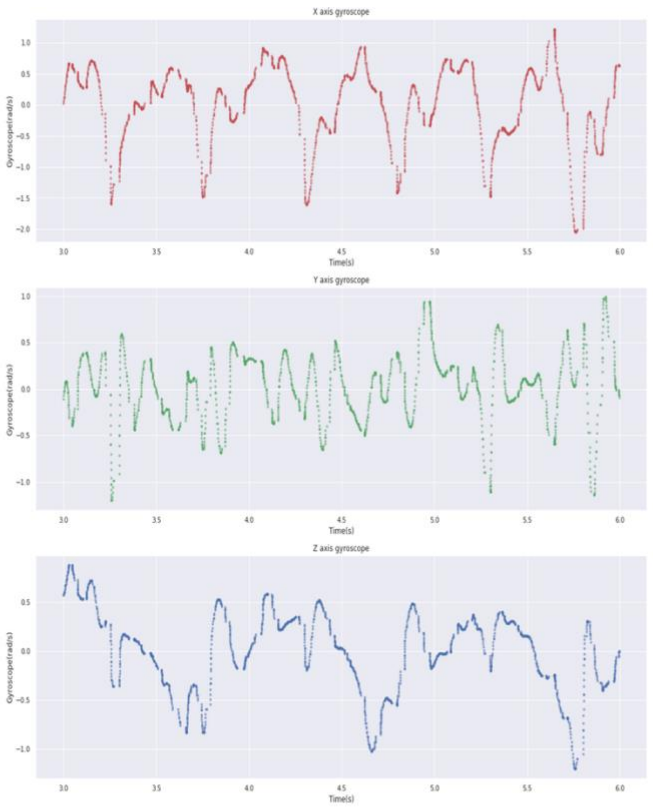
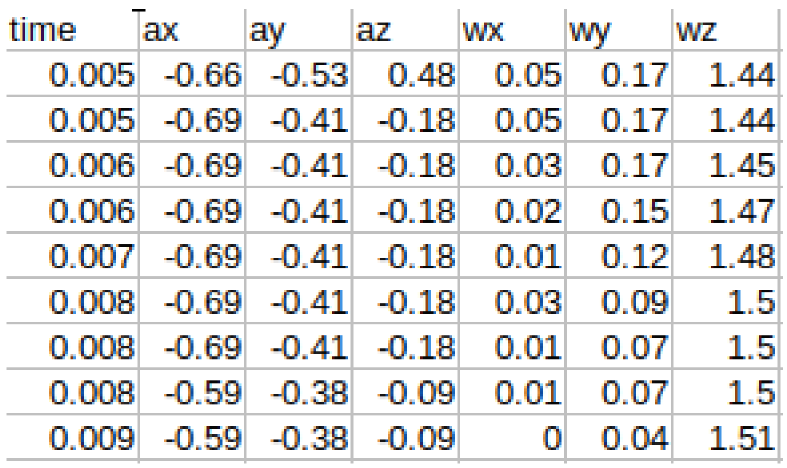


Table IV: Gyroscope Data Sample



time	ax	ay	az	wx	wy	wz
0.005	-0.66	-0.53	0.48	0.05	0.17	1.44
0.005	-0.69	-0.41	-0.18	0.05	0.17	1.44
0.006	-0.69	-0.41	-0.18	0.03	0.17	1.45
0.006	-0.69	-0.41	-0.18	0.02	0.15	1.47
0.007	-0.69	-0.41	-0.18	0.01	0.12	1.48
0.008	-0.69	-0.41	-0.18	0.03	0.09	1.5
0.008	-0.69	-0.41	-0.18	0.01	0.07	1.5
0.008	-0.59	-0.38	-0.09	0.01	0.07	1.5
0.009	-0.59	-0.38	-0.09	0	0.04	1.51

Table V: Raw Data Frame

- **Things about Fall Down Dataset**
  Actually, we are more interested in fall down dataset rather than downstairs/upstairs or walking. Because how to classify fall down state may have more important real-life application for us. Like apple watch, whenever people who wearing it falls down and

not stand up in a certain time, apple watch will send message to emergency contact to ask for help. But for collection of falls down dataset, challenge occurs. After the discussion, we plan to walk 2 seconds and then fall down violently on the mattress. For Data Cleaning part, we manually intercept fall down part to get helpful information.

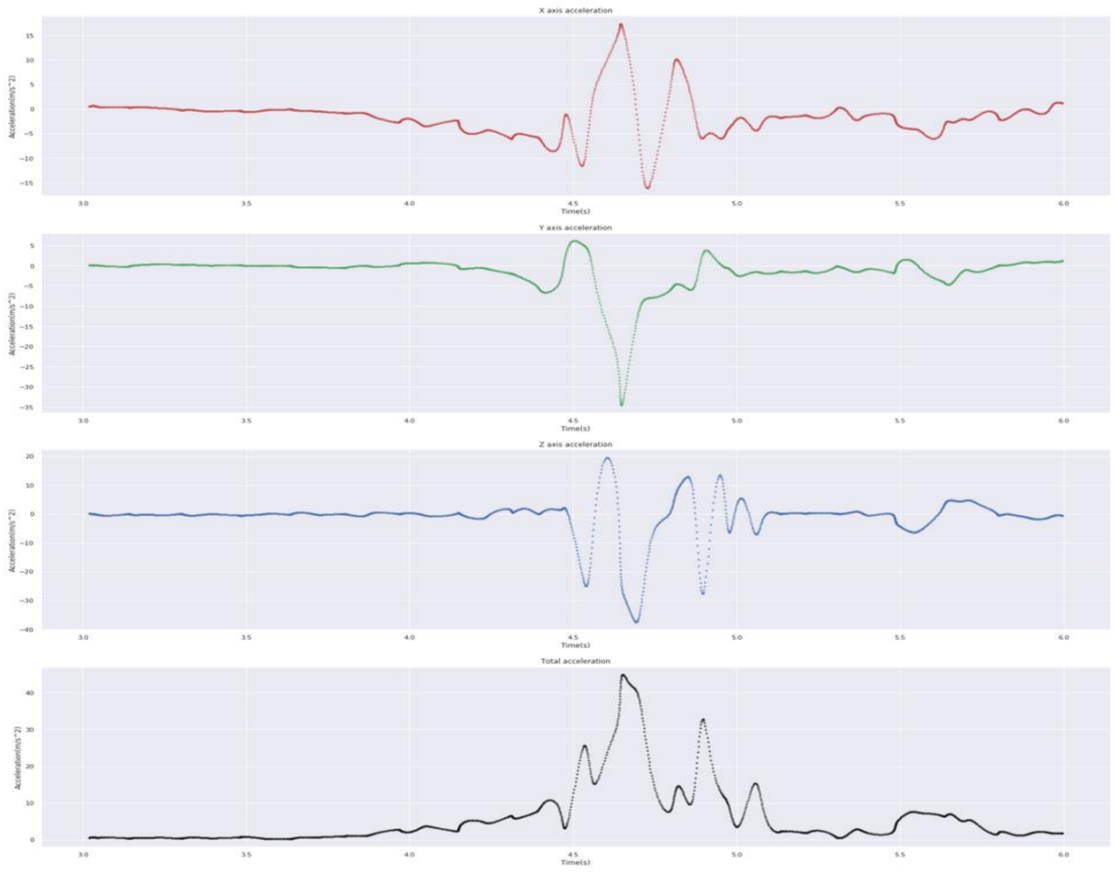


Table VI: Fall Down Data Sample

- **Data Collection Summary**

We used an Android phone with app *Physics Toolbox Suite* by *Vieyra Software* to collect sensor data. We designed data collection under two daily scenarios, walking with the phone in hand and walking with the phone in pocket. For each scenario, we collect 15 sets of data for each category from walking on flat, going upstairs, going down stairs, and falling down. Each data has duration of 6-7 seconds for further data processing and cleaning.

As there are several built-in sensors in the phone, we decided to use linear accelerator and gyroscope to support our project. By physics definition, linear accelerator records the rate of velocity change on each axis (x, y, z), and gyroscope record the angular velocity on each axis (x, y, z). With accelerate on each axis, we can analyze information of people's step frequency which has a periodical circulation feature.

Additionally, gyroscope can give us different data presentation on cell phone angular motion status regard to its axis which can be used to determine whether the phone is in hand or in pocket.

Data Processing and Cleaning

- **Truncate Data**

  After the raw data has been collected, we first truncate all data in first 3 second as it is mainly about the action of putting the phone into the pocket, to make this process easier, we hold the phone in hand without any motion in first 3 seconds when collecting in-hand data.

- **Filtering**

  To clean the data and avoid noise data bias our model, we use Butterworth filter to smooth the walking data including walking on flat, go upstairs and go downstairs. As Butterworth filter has a great effect on outliers out of a certain frequency, we assume walkers have their own steady step frequency and every step has a certain range of data related to our research.

  For fall down data, as the accelerate could have an extremely large absolute value when cellphone holder hit the ground which is a typical feature of distinguishing it from other motion states, so we decide not to use any filter and totally trust data from sensors.

  In Table VII, we can clearly see that the z axis is extremely regular. Comparing with raw downstairs hold dataset, Butterworth have some process deal with missing data and so that raw data set can be shown as a smooth curve.

  Since Butterworth has a great effect on data points far away from a certain range and considering the large accelerate can be an important feature on fall down data, we decide not to use Butterworth filter deal with fall-down-hold and fall-down-in-pocket datasets.

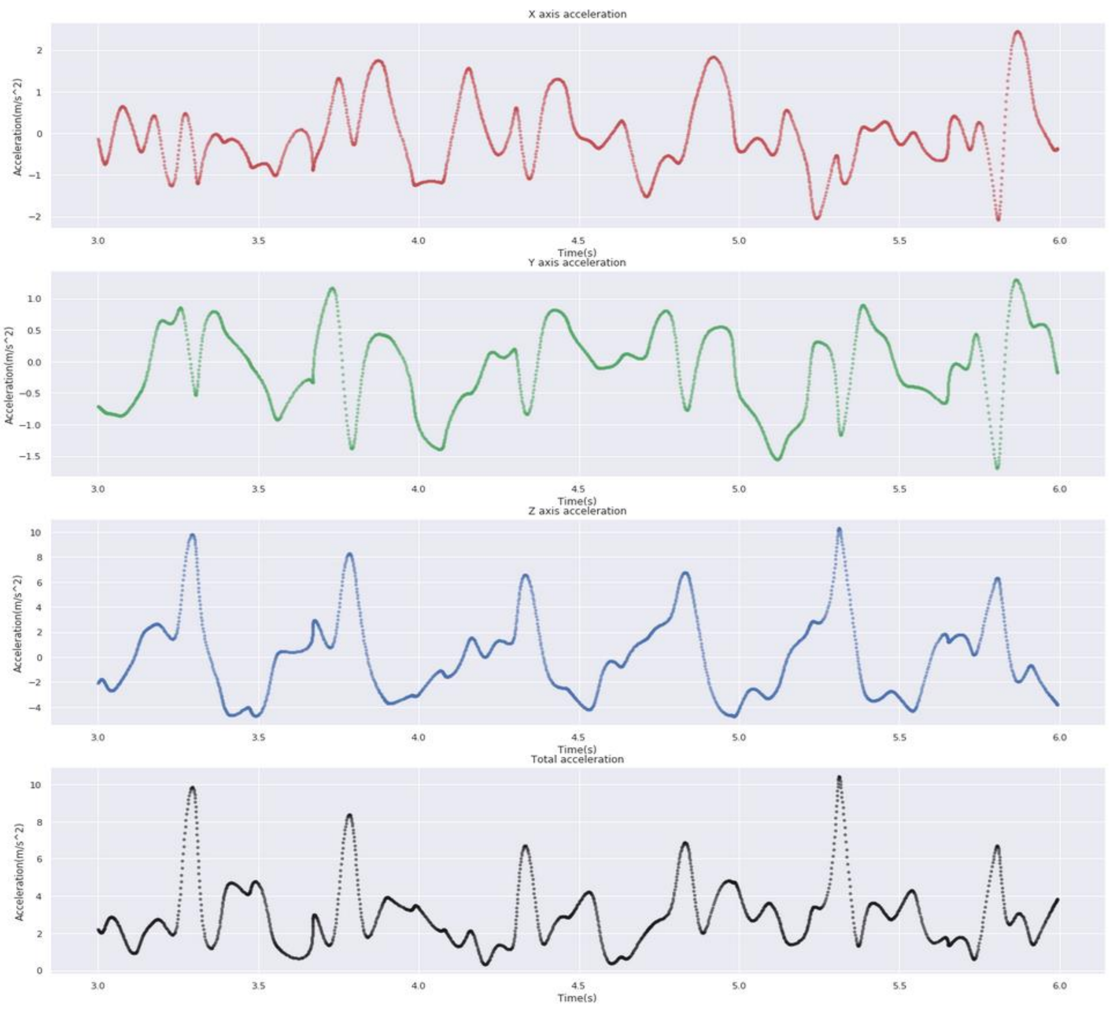


Table VII: Acceleration Data after Butterworth Filtering

Design Feature Structure

- **Basic feature**

  We have more than 2000 points for each dataset and we definitely cannot pass all the points to the Machine Learning model. Therefore, we need to get some features to represent the dataset. According to previous exercises, we use `.describe()` function to get the average, maximum, 75% percentile, median, 25% percentile and minimum of each dataset as its feature. **Moreover, we want to calculate the maximum slope of acceleration which we hope it can distinguish fall down from others' states. However, it didn't perform well when we give it to ML model**. Hence, we discard this feature. In the end, we calculate total accelerate by Euclidean norm ($a_{total} = \sqrt{{a_x}^2 + {a_y}^2 + {a_z}^2}$) and add the result into a new feature. The final data frame is shown in Table VIII below.

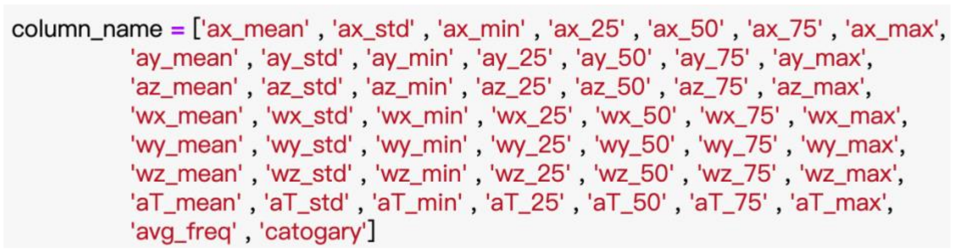

```
column_name = ['ax_mean' , 'ax_std' , 'ax_min' , 'ax_25' , 'ax_50' , 'ax_75' , 'ax_max',
               'ay_mean' , 'ay_std' , 'ay_min' , 'ay_25' , 'ay_50' , 'ay_75' , 'ay_max',
               'az_mean' , 'az_std' , 'az_min' , 'az_25' , 'az_50' , 'az_75' , 'az_max',
               'wx_mean' , 'wx_std' , 'wx_min' , 'wx_25' , 'wx_50' , 'wx_75' , 'wx_max',
               'wy_mean' , 'wy_std' , 'wy_min' , 'wy_25' , 'wy_50' , 'wy_75' , 'wy_max',
               'wz_mean' , 'wz_std' , 'wz_min' , 'wz_25' , 'wz_50' , 'wz_75' , 'wz_max',
               'aT_mean' , 'aT_std' , 'aT_min' , 'aT_25' , 'aT_50' , 'aT_75' , 'aT_max',
               'avg_freq' , 'catogary']
```

Table VIII: Data Frame of Features

- **Fourier Transformation**

  According to the recommendation from professor and combine with online article, Fourier Transformation is a good way to transform data from a time spectrum to frequency spectrum (Han, 2018). We use Fourier Transformation `fft.fft()` from Numpy to get the frequency spectrum of our step frequency with frequency (Hz) on X-axis and number of data points on Y-axis. With the distribution, we can have our analysis on whether this specific subject has a normally distributed step frequency.

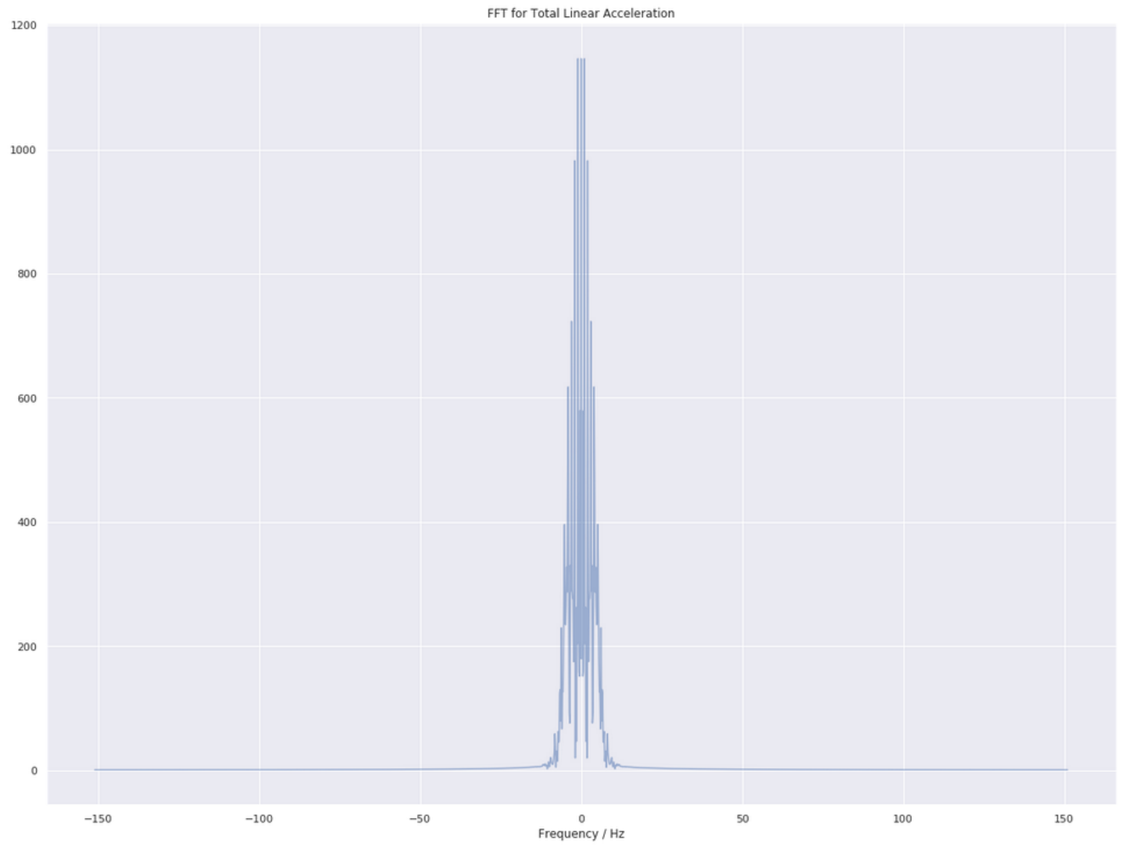


Table IX: Total Linear Acceleration on Downstairs Hold Dataset after FFT

- **Ongoing Test Design Relates to Fourier Transformation:**

  1. Normal test on the FFT graphs to determine whether they are normal distributed or not.

2. If our datasets pass the normal test, use ANOVA and post hoc Tukey test to determine whether different motion states have same step frequency.
3. If the test turns to an unequal result, why does this happen and what is the factor behind this.

Machine Learning Model

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead (Wikipedia, 2019). During the course, we were introduced Naïve Bayes model, KNN model, SVC model, Neural Network and decision tree to do the classify job. In our project, we plan to test whichever model does the best job, and let it classify some new data.

Experiments Process Design

With taking look at the data collected by our teammates, we discovered several interesting facts. Three walking datasets likely have normal distribution, the mean of each distribution tends to be the same. Therefore, we are going to use `.normaltest()` do normal test on each walking dataset, with the p-value from the function to check we can determine whether it is normal distributed or not. If all of the datasets pass the test, we are going to use ANOVA table to find if the means of step frequency are equal.

We use functions from `sklearn` python library to help us complete machine learning. We build `pipeline` for each model to make our code looks neat and concise. After using `train_test_split()` function split our data into training dataset and test dataset, we build naïve Bayes model, KNN model, SVC model, neural network, and decision tree model and pass the dataset into them. With the result from each model, we use `.score()` to compare whichever model does better job. Moreover, we are going to collect additional sets of data to let our model identify collector's motion states.

Results and Analysis

Normal Test

We assume walker's step frequency has a normal distribution during each motion states. We use `.normaltest()` to do normal test on each Fourier transformed dataset, and check p-value. The results we got from these tests are all very close to 0. By the rule of using normal test and statistic hypothesis, if the p-value from test is greater than 0.05, we do not have confidence to say the dataset has a normal distribution. The results from normal test on each dataset are shown in Appendix I.

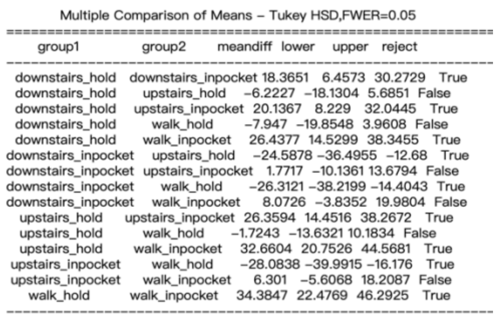
We reviewed the data plot and realize we have more than 1500 data points in each dataset. According to the definition of central limit theorem: the sample means, $\overline{S_k}$, tend toward a normal distribution $\mathcal{N}(\mu, \frac{\sigma^2}{n})$ with large enough n (Baker, 2019). So that we merge our data of each category and going to pass the new dataset into ANOVA table as each dataset is regarded as normal distribution.

ANOVA Test

We passed the merged dataset into `.f_oneway()` function, the p-value came out to be 1.097×10^{-25}, which gives us great confidence to believe the step frequency is not same among motion states we work on. As there exist at least one pair of dataset has different step frequency, we decide to do post hoc analysis to find out whichever pair has different step frequency.

Post Hoc Tukey Test

We use `pairwise_tukeyhsd()` to do the post hoc Tukey test and we got a table of t-test result table. From the post hoc test result in Table X, we find all the data of cellphone hold in hand pair with data of cellphone in pocket has been rejected by the test. Moreover, from the `plot_simultaneous()` shown in Table XI, we can find there exist a clear gap between hold-in- hand data and in-pocket data. When we get this interesting result, we start to pay attention to the difference between people walk with cellphone in hand and with cellphone in pocket. After several days observation we find the reason of the difference may because of people always try to stabilize their cellphone to read or text when holding their phone in hand. Under the control of hand, we assume some vibrate caused by walking can be eliminated and reduce the mean of step frequency.



```
        Multiple Comparison of Means - Tukey HSD,FWER=0.05
=====================================================================
      group1            group2       meandiff  lower    upper   reject
---------------------------------------------------------------------
  downstairs_hold  downstairs_inpocket 18.3651   6.4573  30.2729   True
  downstairs_hold     upstairs_hold    -6.2227 -18.1304  5.6851  False
  downstairs_hold   upstairs_inpocket  20.1367   8.229   32.0445   True
  downstairs_hold       walk_hold      -7.947  -19.8548  3.9608  False
  downstairs_hold     walk_inpocket    26.4377  14.5299  38.3455   True
downstairs_inpocket   upstairs_hold   -24.5878 -36.4955 -12.68     True
downstairs_inpocket upstairs_inpocket  1.7717  -10.1361 13.6794  False
downstairs_inpocket     walk_hold     -26.3121 -38.2199 -14.4043   True
downstairs_inpocket   walk_inpocket    8.0726  -3.8352  19.9804  False
   upstairs_hold    upstairs_inpocket 26.3594  14.4516  38.2672   True
   upstairs_hold        walk_hold     -1.7243  -13.6321 10.1834  False
   upstairs_hold      walk_inpocket   32.6604  20.7526  44.5681   True
 upstairs_inpocket      walk_hold    -28.0838 -39.9915 -16.176    True
 upstairs_inpocket    walk_inpocket    6.301   -5.6068  18.2087  False
     walk_hold        walk_inpocket   34.3847  22.4769  46.2925   True
---------------------------------------------------------------------
```

Table X: Post Hoc Tukey Table

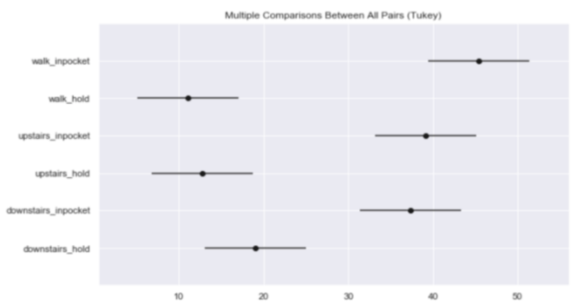


Table XI: Post Hoc Tukey Plot

Machine Learning

- **Training data and validation data**
  Actually, we already get our feature `dataFrame` in Data Cleaning part. In ML part, using `df['category']` as our *y* value and the rest of `dataFrame` part as our X data. Afterwards, call `.train_test_split()` to get X_train , X_valid , y_train , y_valid.

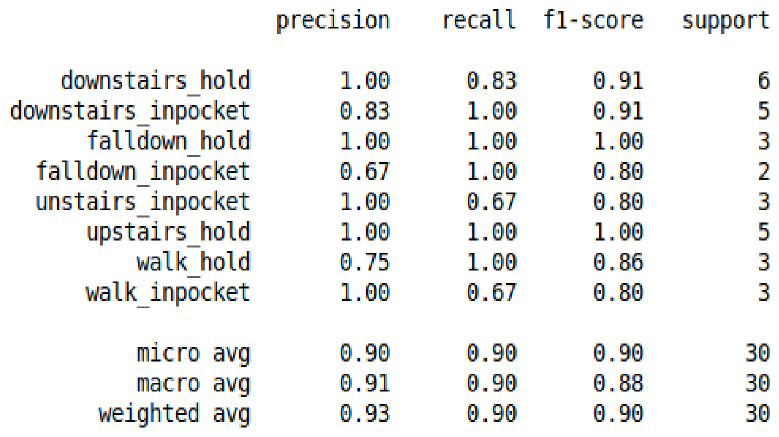
- **Comparing model**
  Note1: According to Baker's advice, high validation data score might not be a good thing. Therefore, we need to also see training data score and classification report whether there is an exceptional situation. For all ML model, we will do such analysis.
  Note2: All scores are from same Training and Validation dataset

- **Naïve Bayes**
  First of all, Naïve Bayes model doesn't have any parameters, so we just make a pipeline to do `StandardScaler` and then fit the model. Surprisingly, Naïve Bayes performs pretty well by giving high validation score.

  Bayesian classifier training score: 0.944

  Bayesian classifier: 0.9

  However, `.train_test_split()` will randomly divide training data and validation data. Therefore, we are trying several times of fitting model. According to note above, training data score is somewhat higher than validation score (not just a little difference). Hence, we are sure that Naïve Bayes model is somewhat fitting the training data rather than predicting (means **overfitted**). On the other hand, we also need to look at **classification report**.



```
                    precision    recall  f1-score   support

    downstairs_hold       1.00      0.83      0.91         6
downstairs_inpocket       0.83      1.00      0.91         5
       falldown_hold       1.00      1.00      1.00         3
   falldown_inpocket       0.67      1.00      0.80         2
   unstairs_inpocket       1.00      0.67      0.80         3
       upstairs_hold       1.00      1.00      1.00         5
           walk_hold       0.75      1.00      0.86         3
       walk_inpocket       1.00      0.67      0.80         3

           micro avg       0.90      0.90      0.90        30
           macro avg       0.91      0.90      0.88        30
        weighted avg       0.93      0.90      0.90        30
```

Table XII: Naïve Bayes Classification Report

From Table XII and according to Baker's advice, we should check **whether a category is being predicted (means there might be a row or category is all zero which is never ever being predicted).** Fortunately, all categories have been predicted which means except the overfit, Naïve Bayes performs pretty well.

- **K-Nearest Neighbors**

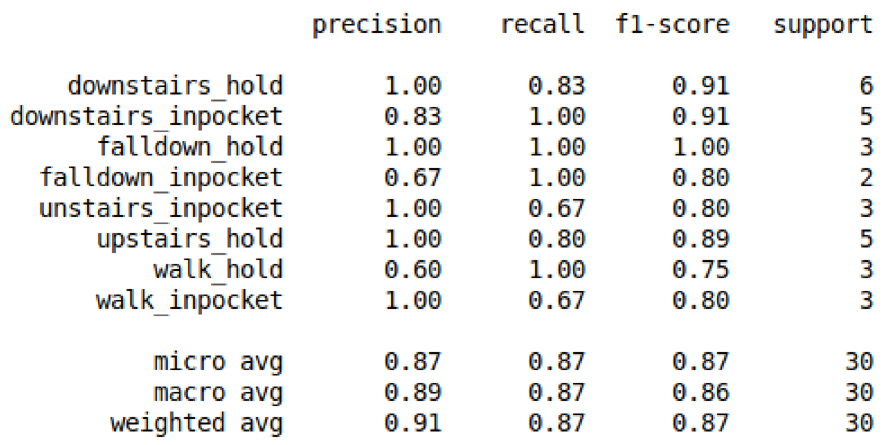
At first, we need to choose how many nearest neighbors to look at which is parameter. After several attempts, we find that there is not too much difference between 3 to 15 for `n_neighbors(parameters)`. Therefore, we choose `n_neighbors` equal to six for kNN model.

Here are the scores:

kNN classifier training score:	0.844
kNN classifier:	0.867

Although kNN model's score lower than Naïve Bayes. However, there is not too much difference between training data score and validation data score which means kNN is predicting rather than overfitting points.

After having a look at the classification report in Table XIII (next page). We can conclude that kNN model better than Naïve Bayes model.



```
                    precision    recall  f1-score   support

   downstairs_hold       1.00      0.83      0.91         6
downstairs_inpocket      0.83      1.00      0.91         5
     falldown_hold       1.00      1.00      1.00         3
 falldown_inpocket       0.67      1.00      0.80         2
 unstairs_inpocket       1.00      0.67      0.80         3
     upstairs_hold       1.00      0.80      0.89         5
         walk_hold       0.60      1.00      0.75         3
     walk_inpocket       1.00      0.67      0.80         3

         micro avg       0.87      0.87      0.87        30
         macro avg       0.89      0.87      0.86        30
      weighted avg       0.91      0.87      0.87        30
```

Table XIII: KNN Classification Report

- **Support Vector Machine Classifier**

We will not mention SVC too much because the training data score is always 1 which is so weird. The classification report in Appendix II.

SVM classifier training score: 1

SVM classifier: 0.833

- **Decision Tree Classifier**

In the last few weeks, we learnt decision tree. Hence, we want to try whether it is useful or not. The classification report is attached in Appendix II.

Decision Tree classifier training score: 1

Decision Tree classifier 0.733

Unfortunately, same as SVC, it just fits the training points rather than predicting. To take a look at what's going on within the decision tree, we can visualize it in Table XIV (next page). In root, we have a condition that if the feature of aT-max is less or equal to 52.12, divided it to downstairs-hold otherwise classifying to upstairs-in-pocket. As a matter of fact, for the leaves at the bottom, there is only one dataset on that category, fifteen datasets on another category. Decision Tree just use one if-condition to divide them which kind of bad.

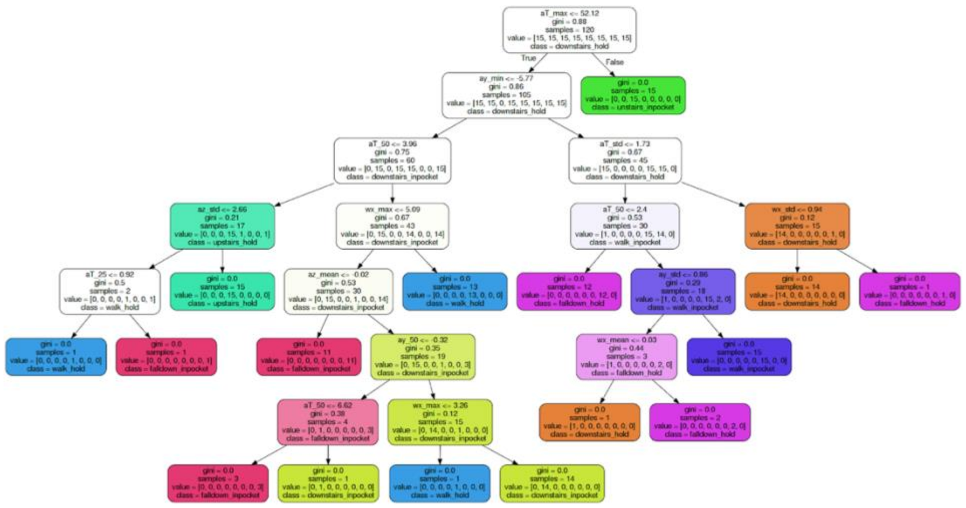

aT_max <= 52.12 gini = 0.88 samples = 120 value = [15, 15, 15, 15, 15, 15, 15, 15] class = downstairs_hold
True
False
ay_min <= -5.77 gini = 0.86 samples = 105 value = [15, 15, 0, 15, 15, 15, 15, 15] class = downstairs_hold
gini = 0.0 samples = 15 value = [0, 0, 15, 0, 0, 0, 0, 0] class = unstairs_inpocket
aT_50 <= 3.96 gini = 0.75 samples = 60 value = [0, 15, 0, 15, 15, 0, 0, 15] class = downstairs_inpocket
aT_std <= 1.73 gini = 0.67 samples = 45 value = [15, 0, 0, 0, 0, 15, 15, 0] class = downstairs_hold
az_std <= 2.66 gini = 0.21 samples = 17 value = [0, 0, 0, 15, 1, 0, 0, 1] class = upstairs_hold
wx_max <= 5.09 gini = 0.67 samples = 43 value = [0, 15, 0, 0, 14, 0, 0, 14] class = downstairs_inpocket
aT_50 <= 2.4 gini = 0.53 samples = 30 value = [1, 0, 0, 0, 0, 15, 14, 0] class = walk_inpocket
wx_std <= 0.94 gini = 0.12 samples = 15 value = [14, 0, 0, 0, 0, 0, 1, 0] class = downstairs_hold
aT_25 <= 0.92 gini = 0.5 samples = 2 value = [0, 0, 0, 0, 1, 0, 0, 1] class = walk_hold
gini = 0.0 samples = 15 value = [0, 0, 0, 15, 0, 0, 0, 0] class = upstairs_hold
az_mean <= -0.02 gini = 0.53 samples = 30 value = [0, 15, 0, 0, 1, 0, 0, 14] class = downstairs_inpocket
gini = 0.0 samples = 13 value = [0, 0, 0, 0, 13, 0, 0, 0] class = walk_hold
gini = 0.0 samples = 12 value = [0, 0, 0, 0, 0, 0, 12, 0] class = falldown_hold
ay_std <= 0.86 gini = 0.29 samples = 18 value = [1, 0, 0, 0, 0, 15, 2, 0] class = walk_inpocket
gini = 0.0 samples = 14 value = [14, 0, 0, 0, 0, 0, 0, 0] class = downstairs_hold
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 0, 0, 1, 0] class = falldown_hold
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 1, 0, 0, 0] class = walk_hold
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 0, 0, 0, 1] class = falldown_inpocket
gini = 0.0 samples = 11 value = [0, 0, 0, 0, 0, 0, 0, 11] class = falldown_inpocket
ay_50 <= -0.32 gini = 0.35 samples = 19 value = [0, 15, 0, 0, 1, 0, 0, 3] class = downstairs_inpocket
wx_mean <= 0.03 gini = 0.44 samples = 3 value = [1, 0, 0, 0, 0, 0, 2, 0] class = falldown_hold
gini = 0.0 samples = 15 value = [0, 0, 0, 0, 0, 15, 0, 0] class = walk_inpocket
aT_50 <= 6.62 gini = 0.38 samples = 4 value = [0, 1, 0, 0, 0, 0, 0, 3] class = falldown_inpocket
wx_max <= 3.26 gini = 0.12 samples = 15 value = [0, 14, 0, 0, 1, 0, 0, 0] class = downstairs_inpocket
gini = 0.0 samples = 1 value = [1, 0, 0, 0, 0, 0, 0, 0] class = downstairs_hold
gini = 0.0 samples = 2 value = [0, 0, 0, 0, 0, 0, 2, 0] class = falldown_hold
gini = 0.0 samples = 3 value = [0, 0, 0, 0, 0, 0, 0, 3] class = falldown_inpocket
gini = 0.0 samples = 1 value = [0, 1, 0, 0, 0, 0, 0, 0] class = downstairs_inpocket
gini = 0.0 samples = 1 value = [0, 0, 0, 0, 1, 0, 0, 0] class = walk_hold
gini = 0.0 samples = 14 value = [0, 14, 0, 0, 0, 0, 0, 0] class = downstairs_inpocket

Table XIV: Visualized Decision Tree

- **Neural Network and Random Forest**

Besides previous models we analyzed above, we additionally tried Neural Network and Random Forest. Those models didn't bring us useful information and we decide to include the details in Appendix II.

Evaluate the Model

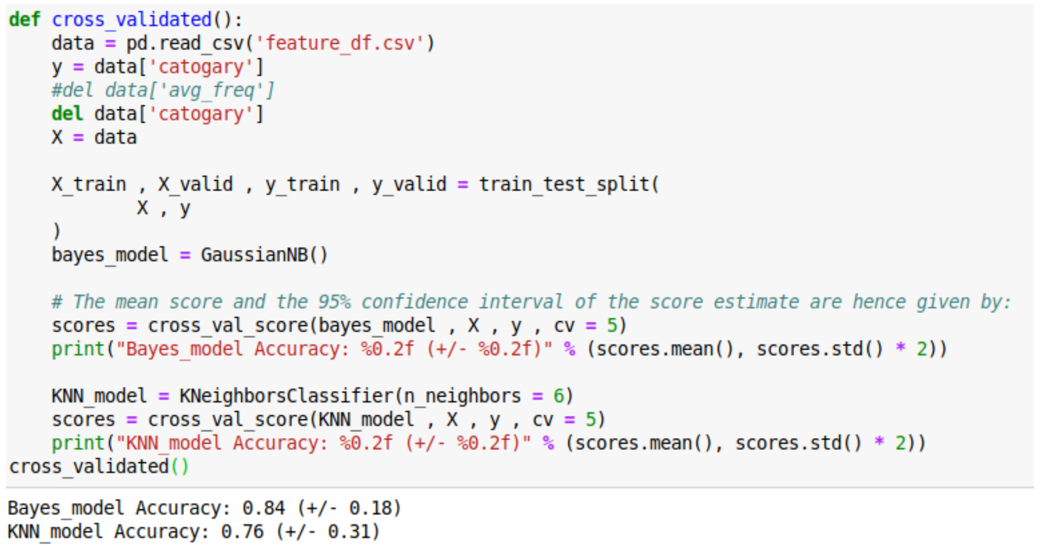
According to the analysis above, we already have two methods to evaluate models

1. Difference between training data score and validation data score
2. Classification Report

Because `train_test_split()` will randomly split the training and validation data. Therefore, we cannot control scores in a certain range. In other words, what's the stats mean of model score. Initially, we do one hundred times of `train_test_split()` and then fit the model to get an average score for each model. Here we only look at Naïve Bayes and kNN throughout previous analysis. Let's see the results.

(0.85, 0.73) -> 0.85 for Naïve Bayes and 0.73 for kNN

However, we get some advice from Baker stated in Table XV (next page).



```
def cross_validated():
    data = pd.read_csv('feature_df.csv')
    y = data['catogary']
    #del data['avg_freq']
    del data['catogary']
    X = data

    X_train , X_valid , y_train , y_valid = train_test_split(
            X , y
    )
    bayes_model = GaussianNB()

    # The mean score and the 95% confidence interval of the score estimate are hence given by:
    scores = cross_val_score(bayes_model , X , y , cv = 5)
    print("Bayes_model Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

    KNN_model = KNeighborsClassifier(n_neighbors = 6)
    scores = cross_val_score(KNN_model , X , y , cv = 5)
    print("KNN_model Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
cross_validated()
```

```
Bayes_model Accuracy: 0.84 (+/- 0.18)
KNN_model Accuracy: 0.76 (+/- 0.31)
```

Table XV: Using cross validated function to get mean of score and 95% confidence interval for each model. Thus, if you want to ask us a certain score for your best model, we can probably conclude that the **mean score of Bayes is 0.84, kNN is 0.76. Moreover, we have a 95% confidence that the mean will fall in range of 0.84 (+/- 0.18) for Bayes, 0.76 (+/- 0.31) for KNN**.

So far, we have demonstrated several Machine Learning Models which we talk about in class. KNN model are absolutely better than all of the other models although it might have a lower score than Naïve Bayes. Naïve Bayes has the best score for our project but it a little bit overfits the training data. SVM, NN, Decision Tree are all worse than models above, so we discard analysis or conclusion we have.

For the reason why our models have various performance, we are going to discuss in Appendix III.

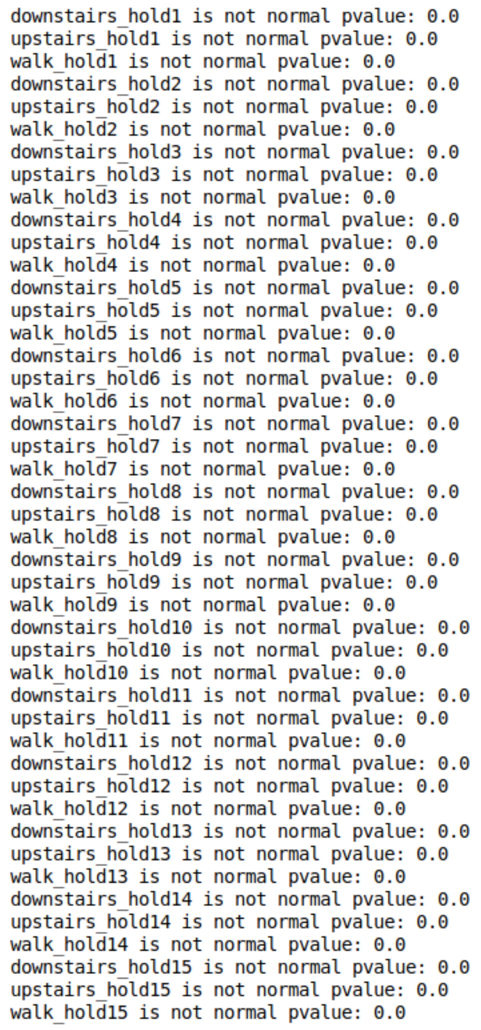
Conclusion

During our group project, we tried out many methods of doing data cleaning, statistic models and worked on some basic machine learning tools to help us classify data from linear accelerate and gyroscope sensor.

From our data and model built, we can conclude people have steady step frequency among walking and go up or down stairs. Moreover, people may try to stabilize their cellphone conscious when walking and texting which makes their step frequency less than usual, but more data and different model should be made to test whether our assumption is true.

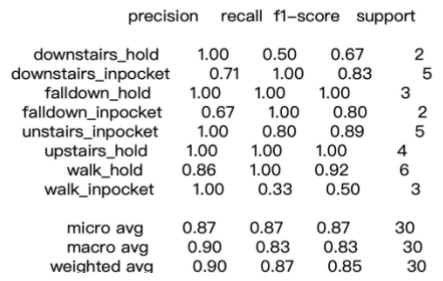
We decide to use naïve Bayes model to build our motion states classifier to determine motion states during a 3-second data. The model does a good job as its training score is 0.944 and validation score is 0.867, also by viewing its classification report, we notice it has the highest f1 score, which indicates it predicts most precisely.

Appendix



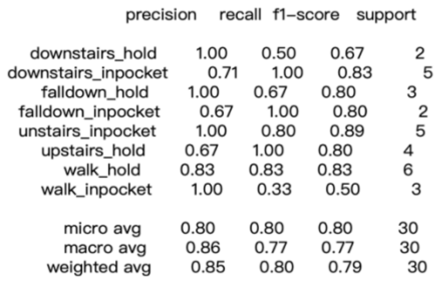
```
downstairs_hold1 is not normal pvalue: 0.0
upstairs_hold1 is not normal pvalue: 0.0
walk_hold1 is not normal pvalue: 0.0
downstairs_hold2 is not normal pvalue: 0.0
upstairs_hold2 is not normal pvalue: 0.0
walk_hold2 is not normal pvalue: 0.0
downstairs_hold3 is not normal pvalue: 0.0
upstairs_hold3 is not normal pvalue: 0.0
walk_hold3 is not normal pvalue: 0.0
downstairs_hold4 is not normal pvalue: 0.0
upstairs_hold4 is not normal pvalue: 0.0
walk_hold4 is not normal pvalue: 0.0
downstairs_hold5 is not normal pvalue: 0.0
upstairs_hold5 is not normal pvalue: 0.0
walk_hold5 is not normal pvalue: 0.0
downstairs_hold6 is not normal pvalue: 0.0
upstairs_hold6 is not normal pvalue: 0.0
walk_hold6 is not normal pvalue: 0.0
downstairs_hold7 is not normal pvalue: 0.0
upstairs_hold7 is not normal pvalue: 0.0
walk_hold7 is not normal pvalue: 0.0
downstairs_hold8 is not normal pvalue: 0.0
upstairs_hold8 is not normal pvalue: 0.0
walk_hold8 is not normal pvalue: 0.0
downstairs_hold9 is not normal pvalue: 0.0
upstairs_hold9 is not normal pvalue: 0.0
walk_hold9 is not normal pvalue: 0.0
downstairs_hold10 is not normal pvalue: 0.0
upstairs_hold10 is not normal pvalue: 0.0
walk_hold10 is not normal pvalue: 0.0
downstairs_hold11 is not normal pvalue: 0.0
upstairs_hold11 is not normal pvalue: 0.0
walk_hold11 is not normal pvalue: 0.0
downstairs_hold12 is not normal pvalue: 0.0
upstairs_hold12 is not normal pvalue: 0.0
walk_hold12 is not normal pvalue: 0.0
downstairs_hold13 is not normal pvalue: 0.0
upstairs_hold13 is not normal pvalue: 0.0
walk_hold13 is not normal pvalue: 0.0
downstairs_hold14 is not normal pvalue: 0.0
upstairs_hold14 is not normal pvalue: 0.0
walk_hold14 is not normal pvalue: 0.0
downstairs_hold15 is not normal pvalue: 0.0
upstairs_hold15 is not normal pvalue: 0.0
walk_hold15 is not normal pvalue: 0.0
```

Appendix I: Normal Test Results



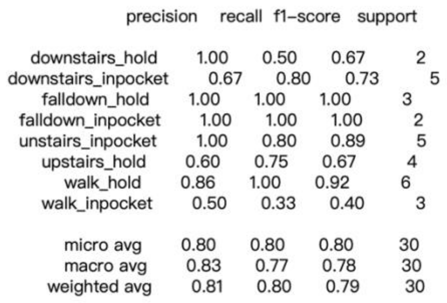
```
                     precision    recall  f1-score   support

    downstairs_hold       1.00      0.50      0.67         2
downstairs_inpocket       0.71      1.00      0.83         5
      falldown_hold       1.00      1.00      1.00         3
  falldown_inpocket       0.67      1.00      0.80         2
  unstairs_inpocket       1.00      0.80      0.89         5
       upstairs_hold      1.00      1.00      1.00         4
          walk_hold       0.86      1.00      0.92         6
      walk_inpocket       1.00      0.33      0.50         3

          micro avg       0.87      0.87      0.87        30
          macro avg       0.90      0.83      0.83        30
       weighted avg       0.90      0.87      0.85        30
```

Appendix II-I: Naïve Bayes Classification Report



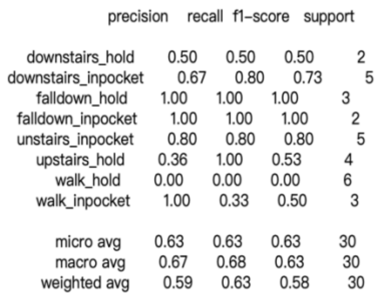
```
                     precision    recall  f1-score   support

    downstairs_hold       1.00      0.50      0.67         2
downstairs_inpocket       0.71      1.00      0.83         5
      falldown_hold       1.00      0.67      0.80         3
  falldown_inpocket       0.67      1.00      0.80         2
  unstairs_inpocket       1.00      0.80      0.89         5
      upstairs_hold       0.67      1.00      0.80         4
          walk_hold       0.83      0.83      0.83         6
      walk_inpocket       1.00      0.33      0.50         3

          micro avg       0.80      0.80      0.80        30
          macro avg       0.86      0.77      0.77        30
       weighted avg       0.85      0.80      0.79        30
```

Appendix II-II: KNN Classification Report



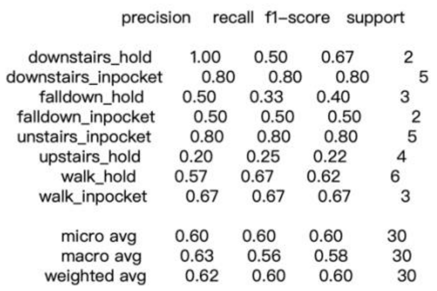
```
                     precision    recall  f1-score   support

    downstairs_hold       1.00      0.50      0.67         2
downstairs_inpocket       0.67      0.80      0.73         5
      falldown_hold       1.00      1.00      1.00         3
  falldown_inpocket       1.00      1.00      1.00         2
  unstairs_inpocket       1.00      0.80      0.89         5
      upstairs_hold       0.60      0.75      0.67         4
          walk_hold       0.86      1.00      0.92         6
      walk_inpocket       0.50      0.33      0.40         3

          micro avg       0.80      0.80      0.80        30
          macro avg       0.83      0.77      0.78        30
       weighted avg       0.81      0.80      0.79        30
```

Appendix II - III: SVC Classification Report



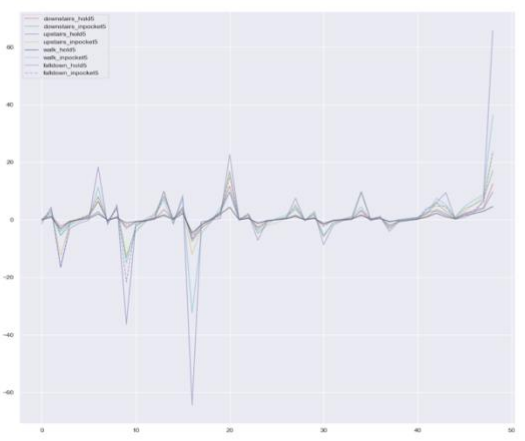
```
                     precision    recall  f1-score   support

    downstairs_hold       0.50      0.50      0.50         2
downstairs_inpocket       0.67      0.80      0.73         5
      falldown_hold       1.00      1.00      1.00         3
  falldown_inpocket       1.00      1.00      1.00         2
  unstairs_inpocket       0.80      0.80      0.80         5
      upstairs_hold       0.36      1.00      0.53         4
          walk_hold       0.00      0.00      0.00         6
      walk_inpocket       1.00      0.33      0.50         3

          micro avg       0.63      0.63      0.63        30
          macro avg       0.67      0.68      0.63        30
       weighted avg       0.59      0.63      0.58        30
```

Appendix II-IV: Neural Network Classification

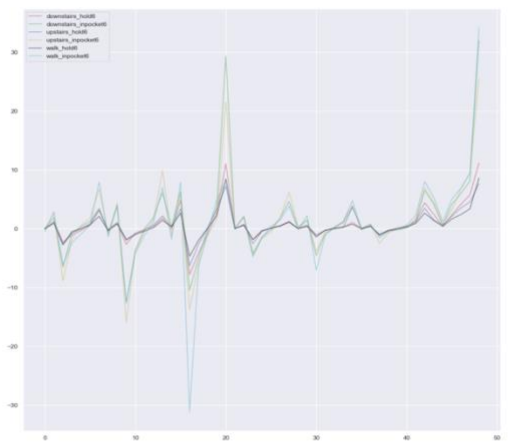


```
                     precision    recall  f1-score   support

    downstairs_hold       1.00      0.50      0.67         2
downstairs_inpocket       0.80      0.80      0.80         5
      falldown_hold       0.50      0.33      0.40         3
  falldown_inpocket       0.50      0.50      0.50         2
  unstairs_inpocket       0.80      0.80      0.80         5
      upstairs_hold       0.20      0.25      0.22         4
          walk_hold       0.57      0.67      0.62         6
      walk_inpocket       0.67      0.67      0.67         3

          micro avg       0.60      0.60      0.60        30
          macro avg       0.63      0.56      0.58        30
       weighted avg       0.62      0.60      0.60        30
```

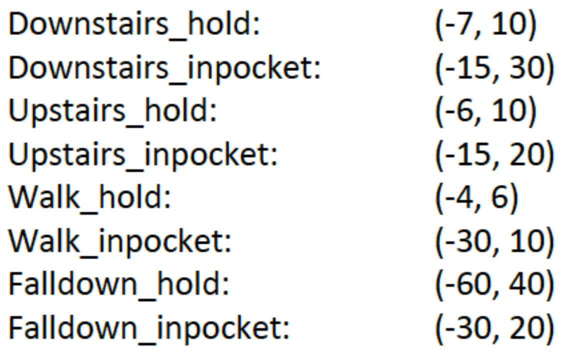
Appendix II - V: Decision Tree Classification



Appendix III-I: Visualized Feature Dataset on Each Category



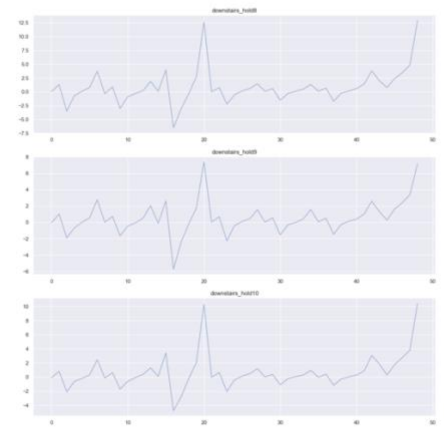
Appendix III-II: Visualized Feature Dataset on Each Category Except Fall-Down

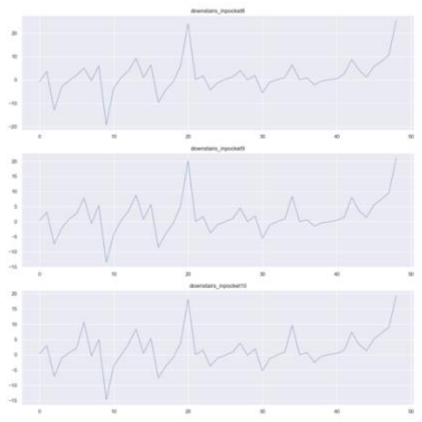


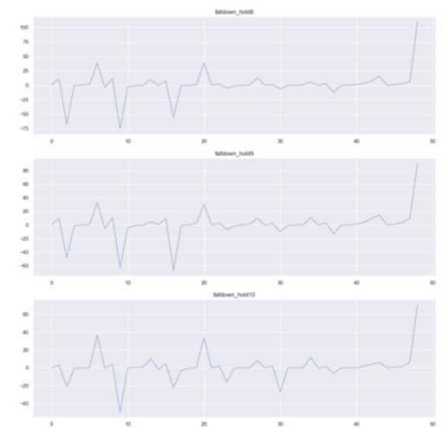
Downstairs_hold:	(-7, 10)
Downstairs_inpocket:	(-15, 30)
Upstairs_hold:	(-6, 10)
Upstairs_inpocket:	(-15, 20)
Walk_hold:	(-4, 6)
Walk_inpocket:	(-30, 10)
Falldown_hold:	(-60, 40)
Falldown_inpocket:	(-30, 20)

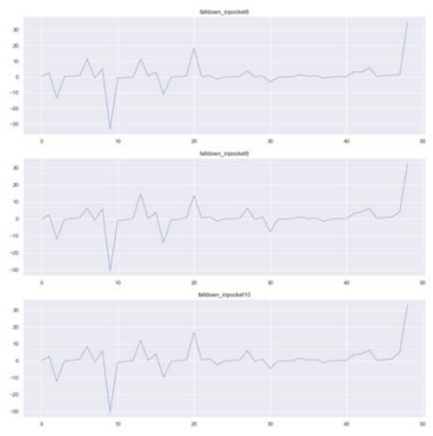
Appendix III-III: Range of Different Features on each category

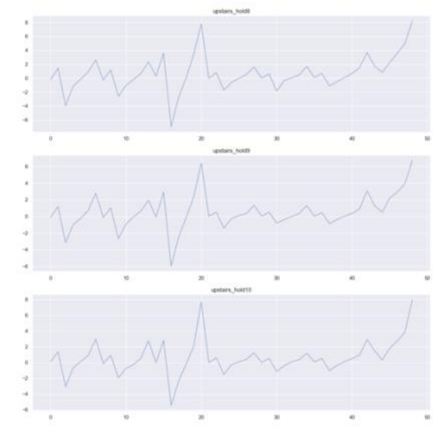
Taking look at the features in each dataset and comparing with other categories (Appendix III-I), we can find there is some outstanding feature in specific category, say, fall down data always have lowest 16th and highest 20th category, which are minimum z-axis accelerate and maximum accelerate. As the value is too large and makes the graph lose details, we decide to remove fall-down data from comparing. In Appendix III-II, we find walk-in-pocket always have minimum z-axis accelerate and downstairs-in-pocket always have maximum z-axis accelerate. The obvious feature can be the major reason of the model perform well. However, there are several categories have similar value range (Appendix III-III), it can be hard for model to distinguish those kinds of category. Three datasets with cell phone hold in hand have range (-5, 10). Thus, the model may get confounded. To have a closer look at each category data feature, we present the plots in Appendix III-IV on next page.

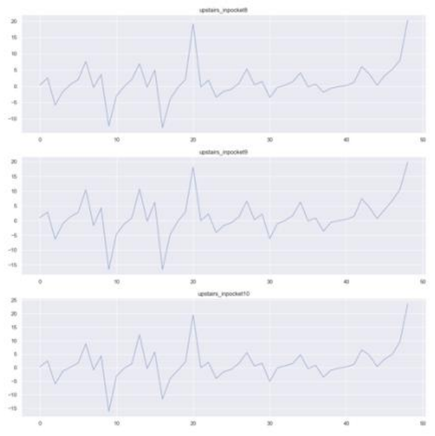




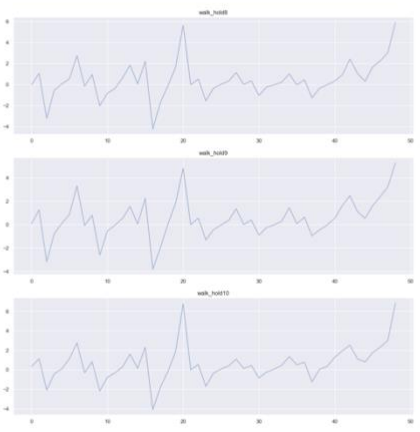


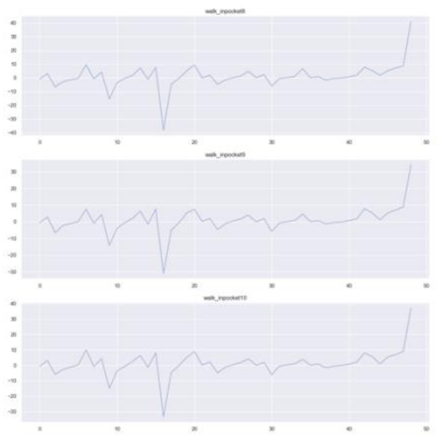






Appendix III-IV





Appendix III-IV Continue

Reference

BakerGregory. (2019). Statistical Tests. Retrieved from: Coursys: https://ggbaker.selfip.net/data-science/content/stats-tests.html

HanHao. (2018 年 Feb 月 2 日). 深入浅出的讲解傅里叶变换. Retrieved from: CNblogs: https://www.cnblogs.com/h2zZhou/p/8405717.html

MalyiViktor. (2017 年 Aug 月 13 日). Run or Walk (Part 2): Collecting Device Motion Data the Right Way. Retrived from: Towards Data Science: https://towardsdatascience.com/run-or-walk-part-2-collecting-device- motion-data-the-right-way-58a277ff2087

Schimid College. (2019). *M.S. Computational & Data Sciences.* Retrieved from Chapman.edu: https://www.chapman.edu/scst/graduate/ms-computational-science.aspx

Wikipedia. (2019, July 20). *Machine learning - Wikipedia.* Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Machine_learning

Project Experience Summary

Hang Peng:

- In order to collect fall down data, carelessness broken his arm and missing CMPT307 midterm. **Thanks for Ben here!!**
- Use SFU GitLab for documentation and version control
- Utilize pandas learned in class to find out correlation among criteria for movie success
- Applied RandomForestClassifier to do multilabel classification for the features of the sensor data, and analysis why RandomForestClassifier perform bad.
- Applied Cross Validated function to evaluate the model and solve the question “Which model perform best?”
- Overviewing and putting together work of teammates to make overall analysis

Bowen Wang:

- Initial project idea and data collection for fall down and walk hold
- Come to a really good idea about focus on fall down datasets and using several Machine Learning tools to analysis fall down datasets
- Collaborating, brainstorming with other two teammates about how to implement the whole project and have effort to find answers for questions, as well as coming up with the overall structure of the program
- Performed ETL processes on several data sets using python tools such as NumPy and pandas.
- Utilize pandas learned in class to find out correlation among criteria for movie success.
- Analyze the Fourier Transform part by using lots of statistical tools.

Duo Lu:

- Initial general data cleaning and manipulating the data for easy analysis
- Brainstormed with another teammate to produce creative questions and solutions
- Collaborate with two teammates to build a project that investigates the relationship between walking and downstairs dataset. By analyzing the walking data, founding why Machine Learning will predict the wrong category.
- Overviewing and putting together work of teammates to make overall analysis
- Visualize and analyze the Decision Tree, figure out the difference between Decision Tree and SVM. As well as the question “Why training data score for DT and SVM is always 1”
- Coding the Fourier Transform part by using NumPy and pandas.