Body Activity Monitoring:

Based on Sensors, Noise, and Walking

CMPT 353: Computational Data Science

Arpit Kaur | 301367803

Sharjeel Ahmad | 301307811

Srimalaya Ladha | 301367941

SFU

1. Introduction

This project is about applying the concepts and techniques covered in the course as relevant to the topic. Our topic is body activity monitoring. The idea behind this project is to predict based on a user's activity whether they are standing, walking, or running.

As we learnt throughout the course, the common collection of steps that are necessary in data analysis pipeline include:

1. Figuring out the question.
2. Acquiring relevant data.
3. Cleaning & preparing the data.
4. Analyzing the data.
5. Interpreting & presenting results.

One of the big questions concerning the course is what do we do after getting the data.

So, the first and the obvious thing to do was to acquire data. To get the data into a nicer format and move from there, we used the ETL technique (extract-transform-load).

We first extracted the data: taking it from the format we found it in (which was a csv format) and reading it.

Then we applied data filtering and cleaning (i.e data pre-processing) and then we saved the data into clean csv files so that we can load it into the next pipeline step. We imported the required libraries(pandas, numpy, matplotlib, scikit learn) at the beginning of each file.

To perform our analysis,we applied statistical tests after calculating mean, min, and max values of required acceleration data.

The interpretation of the results are given through this report.

2. Acquiring Relevant Data

Getting Data

For our purpose, we required data for various activities which varied in movement. We acquired the data by self-collecting it from iOS phone sensors. An app called **"Physics Toolbox Accelerometer"** which collects data from all the sensors on an iPhone was used for this purpose. It recorded data in CSV files.

Data was collected with four variations namely Standing, Walking, Running, and Mixed. Readings for each of the above were acquired across two different modes i.e. by placing the phone in pocket and on the

ankle.Each individual collected 10 files for each activity and after joining them, we had a total of 30 files for each of the three explicit activities.We stored them as raw data. The reason for such diversity in data is because we took upon the advice of the professor to have datasets with varying conditions.We observed that when the phone was placed on the ankle, there was more noise. To deal with that noise, we did signal processing which is described later in the report. After getting the data, we proceeded to clean it.

3. Cleaning/Preparing the Data

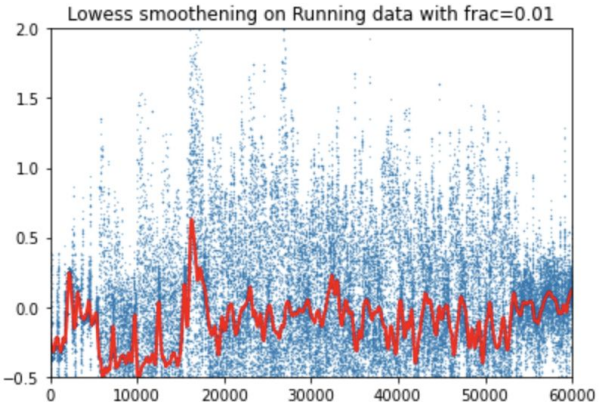
The first thing that came into our mind while cleaning the data was to see if there were any null values in the dataframes, hence we appended the dataframes and looked for null values. If there were any null values, we would have either dropped them or replaced them with some different values like mean depending on the data. Since there were no null values, we did not do any of the above.

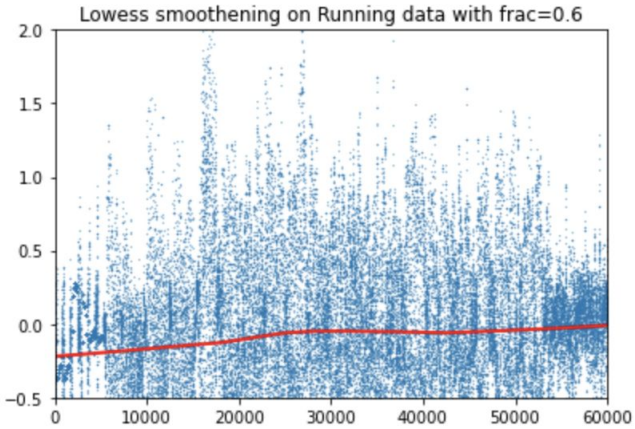
Whenever a value is measured, some error is introduced by the capture or the transmission. This error is known as noise. The measured values in the data are a combination of true values and noise. However, we were just interested in true values(with as little error as possible) and hence we applied filtering algorithms to do signal processing.

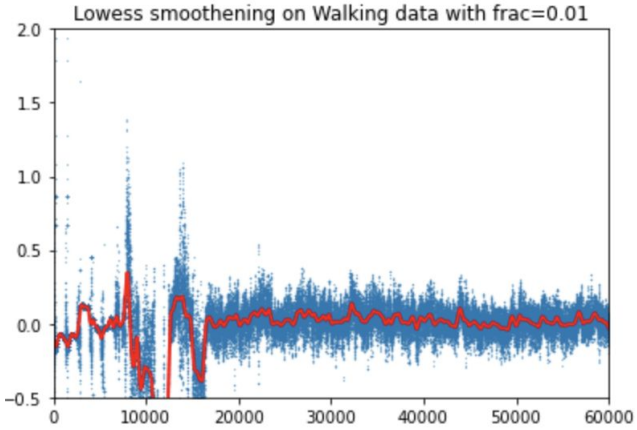
Here, we approximately had 50,000 rows of data for running, 85000 for walking, and 74000 for standing. Our task was to process them.

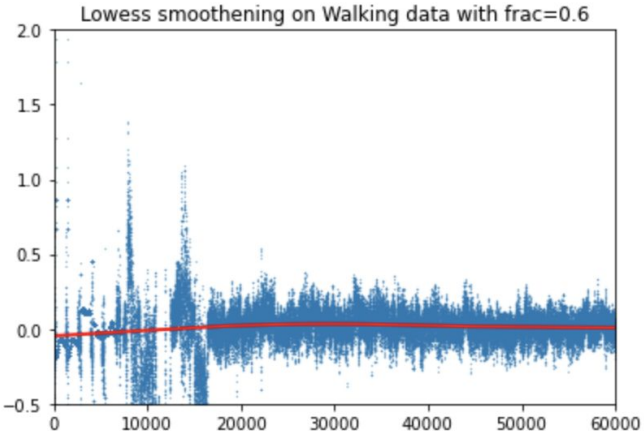
Lowess Smoothing

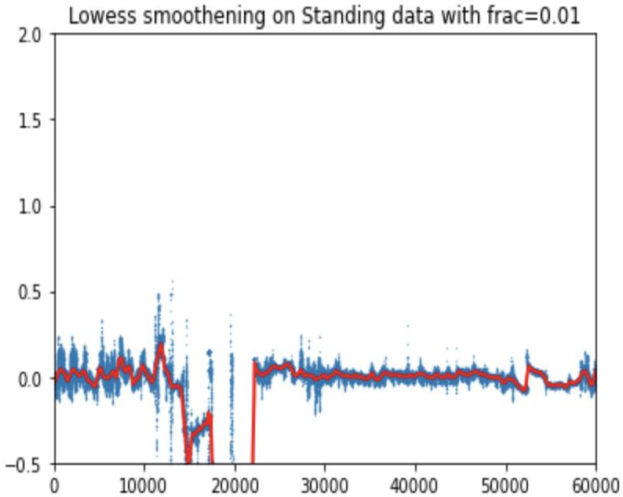
To analyze the noise in the data, we applied a Lowess filter and changed the fractions.The idea behind this was that we had a lot of data and Lowess smoothing tends to work good with it. We observed that the Lowess filter either over-smoothes the data(when the fraction is taken to be large) or becomes over-sensitive to noise (when a smaller fraction is taken). One of the plots that we get after applying Lowess filter on x acceleration of all datasets:

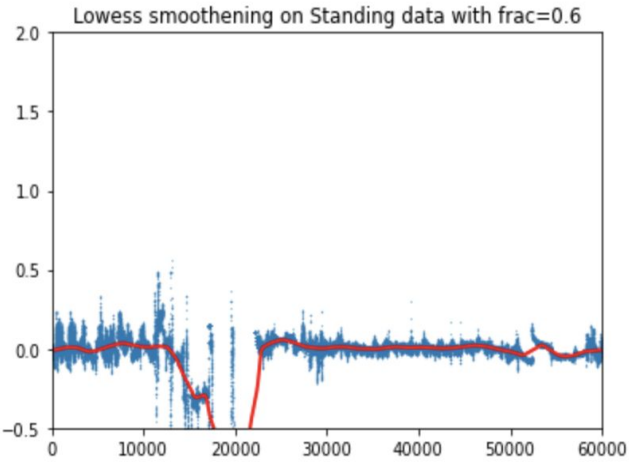

Lowess smoothening on Running data with frac=0.01
2.0
1.5
1.0
0.5
0.0
−0.5
0
10000
20000
30000
40000
50000
60000


Lowess smoothening on Running data with frac=0.6
2.0
1.5
1.0
0.5
0.0
−0.5
0
10000
20000
30000
40000
50000
60000


Lowess smoothening on Walking data with frac=0.01
2.0
1.5
1.0
0.5
0.0
−0.5
0
10000
20000
30000
40000
50000
60000


Lowess smoothening on Walking data with frac=0.6
2.0
1.5
1.0
0.5
0.0
−0.5
0
10000
20000
30000
40000
50000
60000


Lowess smoothening on Standing data with frac=0.01
2.0
1.5
1.0
0.5
0.0
−0.5
0
10000
20000
30000
40000
50000
60000

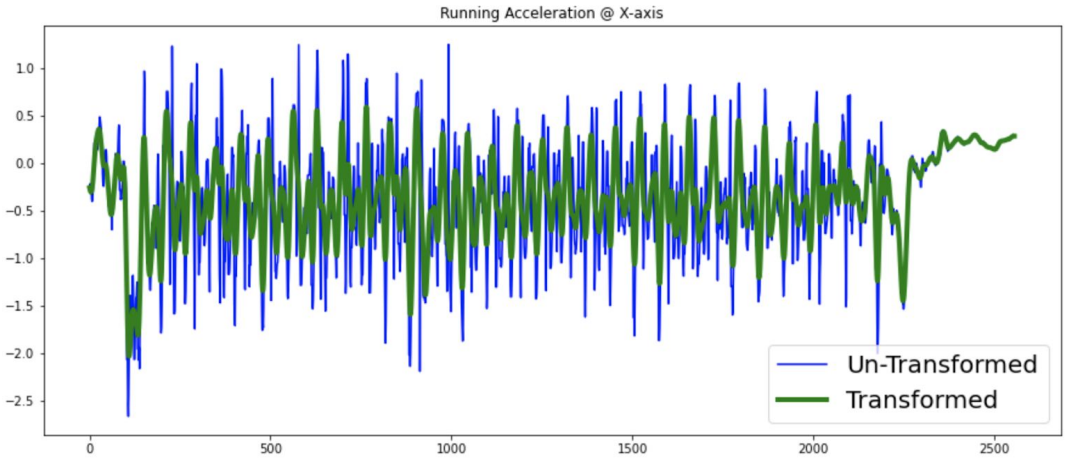

Lowess smoothening on Standing data with frac=0.6
2.0
1.5
1.0
0.5
0.0
−0.5
0
10000
20000
30000
40000
50000
60000

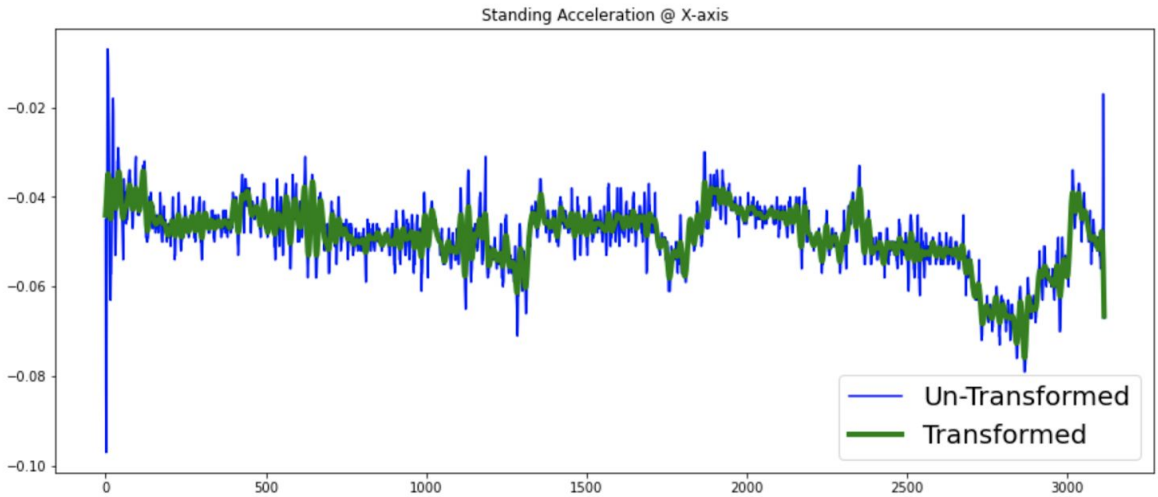
Butterworth Filter

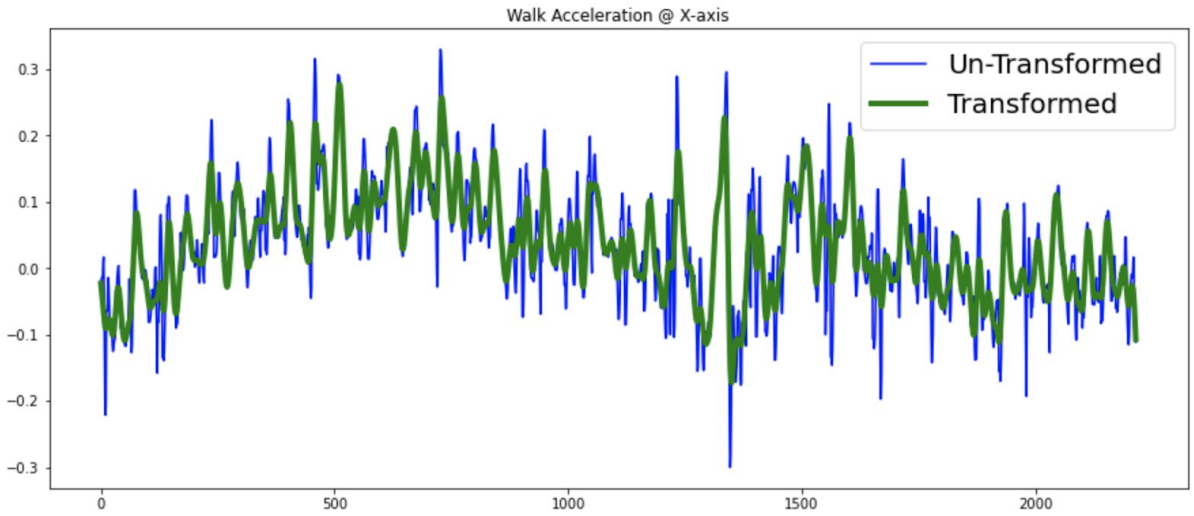
After realizing that Lowess smoothing would not work, we applied a low pass filter using the Butterworth filter.A low pass filter keeps the low frequencies and discards the high. Since our data was recorded in real-life conditions and on a phone, it was bound to have a lot of noise.Random noise adds high frequency signals to the data. Butterworth filter reduces noise by ignoring any outliers within the frequency. For our purpose, we used a digital filter with a signal order of 3 and a frequency cap of 0.1(i.e it keeps the frequency <0.1 which means > 10 samples/cycle). If we keep the frequencies too high, noise will get through. If we keep them too low, we can lose frequencies that are real signal.

The reason for choosing a digital filter was the suggestion of the professor. Also, it is a good filter.

Here is how our data differed with and without the low-pass filter:







Feature Engineering

The definition of feature engineering tells us that it is a process of using domain knowledge of the data to create features that make algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process.

The kind of feature values we have for a dataset have a huge effect on what we can analyze and predict about the data. We had a total of 90 datasets(30 each). Each dataset has about 2000 rows and this means that for a single type of data (say running), we have about 60000 rows.A large dataset requires lots of processor memory. It is often hard to learn those many parameters. Hence we aggregated the data by taking the mean values from each dataset. Along with the mean, we took maximum and minimum values so that we can get to know the range in which the respective datasets lie. It gives us fewer rows to work with. We have chosen this kind of aggregation because the values in our datasets are numerical and it makes sense to use the mean values.

We used basic functions for finding these values for individual files and at the end, we appended all the values of mean, minimum and maximum(for acceleration on x ,y and z axis respectively) into one pandas dataframe.

We saved the final appended dataset in a csv file so that we can use it later while applying machine learning algorithms.We also used them while doing statistical analysis and to see if we have any outliers. The application is described later in the report.

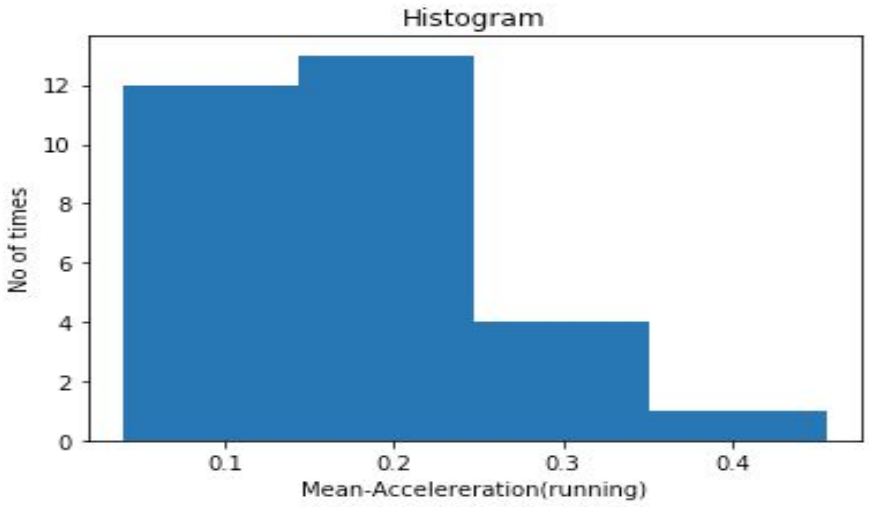
4. Analyzing the data.

Statistical analysis

After applying the basic statistics we had data frames for “Running”, “Walking”, and “Standing”. Each data frame contains mean values for different dimensions, Maximum and Minimum values for different dimensions, and the magnitude of total acceleration before and after transformation. For our statistical analysis we will be using the magnitude of total acceleration after transformation that goes by the column name “mean1Acc-turns ” in our data frames.

- **Normality:**

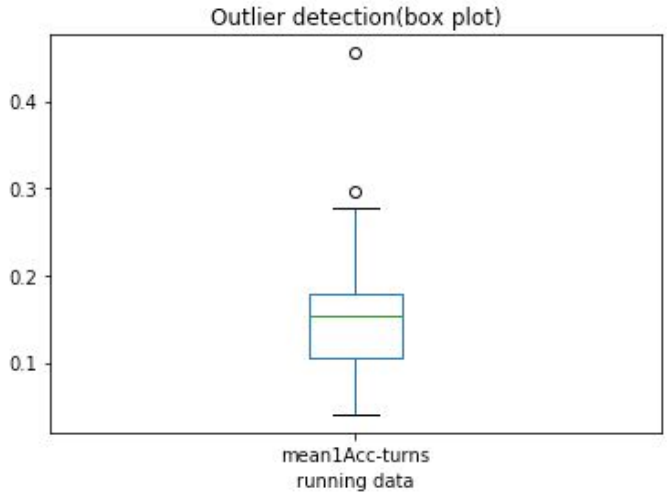
We drew some basic plots to get a sense of the data. Histograms were drawn using that mean magnitude for total acceleration to check if the data points are normally distributed or skewed. In the three scenarios “Standing”, “Walking” and “Running” the data points showed right skewness. This basically implies that the mean for our data points was greater than the median. We could have applied some Transformations to convert right skewed data into Normal distribution to apply statistical tests such as t-test because it assumes normal distribution but in our case we have more than 2 groups so t-test is not applicable. More statistical tests will be discussed in the statistical tests section of the report. The sample Histograms showing data distribution for “running” is shown below:



- **Outliers detection**

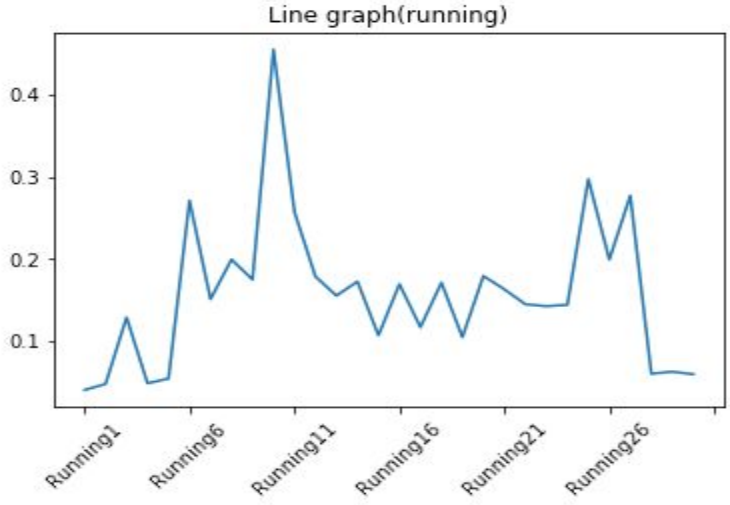
We drew box plots to check if some points are significantly different than others and mark them as outliers. In our analysis all three categories showed some outliers but we decided to keep the outliers

instead of removing them as we deem them important for our analysis/classification purposes. The reason for keeping them was because of the fact during the data collection the pace of "walking" and "running" was different as the data was collected by different members of the group. The sample box plot from the running data is shown below.



- **Peak points**

We also drew some line graphs to look at peak points during "Running", "Walking" and "Standing" to check which particular instance was higher than the others. The results were kind of expected as "Standing" peaks were damped and showed lower values. The peaks for "Running" data showed abrupt increase and decrease depending on the running speed. The "Walking" data showed peaks that were very close to each other and implied consistency. The sample line graph for running data is shown below.



- ## Statistical Tests

i. T-test

T-test is not applicable in our case because the data is not normal and we have more than 2 groups in our analysis.

ii. ANOVA

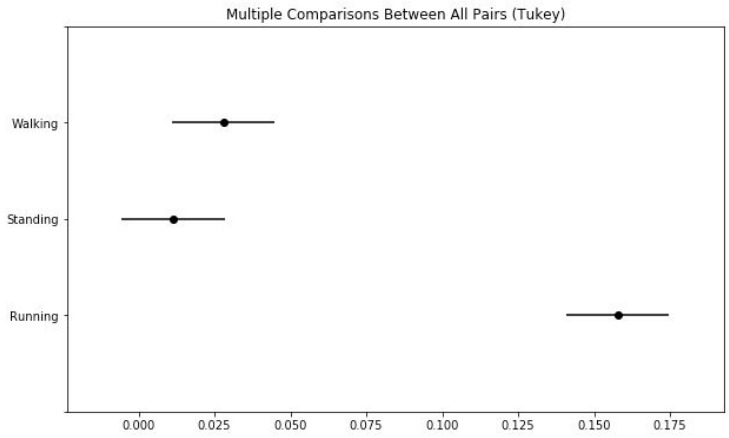
Analysis of Variance was used to check if there is a difference between mean values of magnitudes of acceleration for "Standing" , "Walking" and "Running". The p-value for our analysis turned out to be less than 0.05 which implies that there is some difference in the means of these groups but ANOVA does not tell us that which two groups out of our combinations have different means but ANOVA results showed statistical significance.

iii. Post-Hoc Analysis

We got statistical significance in ANOVA($p<0.05$) so it makes sense to do Post-Hoc Analysis as well. This will help us check which two groups had different means. We just have 3 groups. It will result in 3 combinations according to the formula nc2. In the figure below if the "reject" column shows "False" then that implies means are the same for these 2 groups whereas the opposite is true where the reject column shows "True". For the combination of "Standing" and "Walking" the means were the same whereas the means were different for the other 2 combinations. The comparison table is shown below.

Multiple Comparison of Means - Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
Running	Standing	-0.1464	0.001	-0.1802	-0.1126	True
Running	Walking	-0.1298	0.001	-0.1636	-0.0959	True
Standing	Walking	0.0166	0.4757	-0.0172	0.0504	False



iv. Chi-squared test

We do not have Categorical data so Chi-square test was not applied.

v. Mann Whittney U test

Mann Whittney test was applied as we were not sure about the true underlying distribution of the data. The Mann Whittney test will tell us if the data from one group was larger/smaller than the other or not. The results ($p<0.05$) show that the values for the most of our groups tend to sort higher than the others which basically implies that values in the groups our combinations are not equally shuffled.

Machine Learning

After doing the feature engineering we had features that could be used for Machine learning. We will be using the classification aspect of machine learning for our analysis. Based on the engineered features given to the classification model it should classify them as "Standing" , "Running" or "Walking".

Engineered Features:

The engineered features that will be used for training the model are shown in the data frame below.

mean1AccX	mean1AccY	mean1AccZ	mean1Acc-old	mean1Acc-turns	min1AccX	min1AccY	min1AccZ	min1Acc-old	min1Acc-turns	max1AccX	max1AccY	max1AccZ
0.020516	-0.332795	-0.934976	1.000219	0.014013	-0.127838	-0.494107	-1.201768	0.829	2.395191e-07	0.218681	-0.070369	-0.790042
-0.008327	-0.233387	-0.966295	0.998388	0.009913	-0.120362	-0.371054	-1.054567	0.930	6.179249e-06	0.183071	-0.054697	-0.890094
0.021418	-0.223114	-0.969937	0.999033	0.010978	-0.140522	-0.392657	-1.138464	0.798	3.015521e-06	0.300409	-0.066126	-0.793677
0.046227	-0.237113	-0.964393	0.998490	0.017026	-0.105969	-0.341649	-1.057231	0.870	6.297414e-05	0.231294	-0.126824	-0.848331
0.009831	-0.225912	-0.970790	0.998825	0.007442	-0.084022	-0.352971	-1.062356	0.892	1.410546e-06	0.159574	-0.128270	-0.872152
...	...	...	...	...	...	...	...	...	...	...	...	...
-0.113283	-0.294011	-0.988813	1.333722	0.199340	-1.839045	-1.053709	-3.523188	0.148	6.895799e-06	1.353416	1.152213	1.236943
-0.198153	-0.221888	-0.988692	1.455920	0.277106	-1.834903	-1.420457	-3.988615	0.194	9.892599e-05	1.142537	1.173589	1.671844
0.011162	-0.274672	-0.950157	1.119570	0.060195	-1.023355	-0.920399	-3.504760	0.103	7.763990e-07	0.797313	0.374588	1.304387
0.006036	-0.243144	-0.958020	1.056782	0.062569	-1.030350	-1.091835	-2.463197	0.056	1.787004e-06	0.862325	0.327372	0.604819
-0.003130	-0.212008	-0.958579	1.080482	0.059600	-1.019443	-0.867040	-3.103758	0.069	4.253281e-06	0.990463	0.531715	0.556515

rows × 15 columns

Data Splitting:

- The default ratio of 80/20 was used for splitting. 80 percent data was for training the model and 20 percent data was used for the validation set.

Model training:

- We used 7 different models for classification. It was made sure that appropriate parameters are used

to maximize the validation score. Some classifiers were used along with scaling such as "MixMaxScaler" with "SVC" for better results. The data required for this part was stored as a folder labelled "readyToTrain".

1.Random Forest

2. KNeighbors

3. Naive Bayes (Gaussian)

4. Neural Nets (MLPClassifier)

5. Gradient Boosting

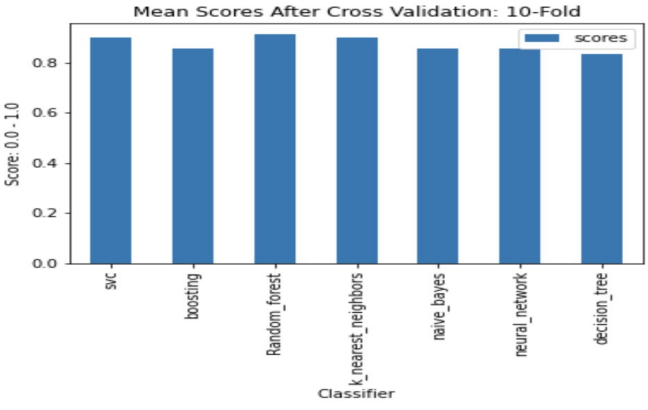
6. SVC

7. Decision Tree

Model Evaluation Methods:

1. **K-folds Cross Validation:**
   - When data is divided into different sets(Training and Validation) the number of samples used for training the model are reduced and results can depend on the particular random choice for the pair of sets which is not good. This problem is solved by k-folds cross validation and helps in evaluation performance of the model based on the validation scores. We used 10-folds cross validation for our analysis which implies that
     - Model is trained using 9 of the folds as training data.
     - The resulting model is evaluated on the remaining part of the data.
   - We applied 10-folds cross validation to every model and took mean of the validation scores and then plotted a bar graph of the scores to check which model gives the best mean validation score. The mean validation scores and bar graph are given below. The **random forest classifier** seems to give the best results.

```
K-fold Mean Scores:
     scores            classifier
0  0.900000                   svc
1  0.855556              boosting
2  0.911111         Random_forest
3  0.900000  k_nearest_neighbors
4  0.855556           naive_bayes
5  0.855556         neural_network
6  0.833333         decision_tree
```



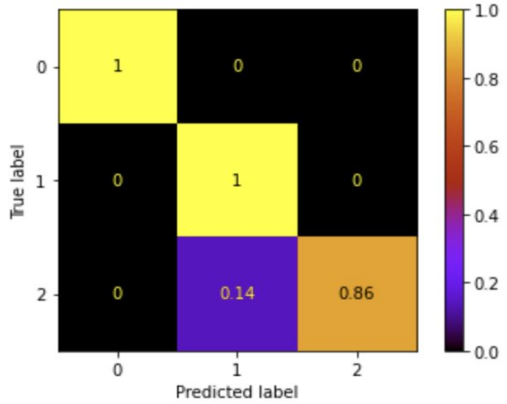
2. **Evaluation based on Classification report, Confusion matrix and ROC curve:**

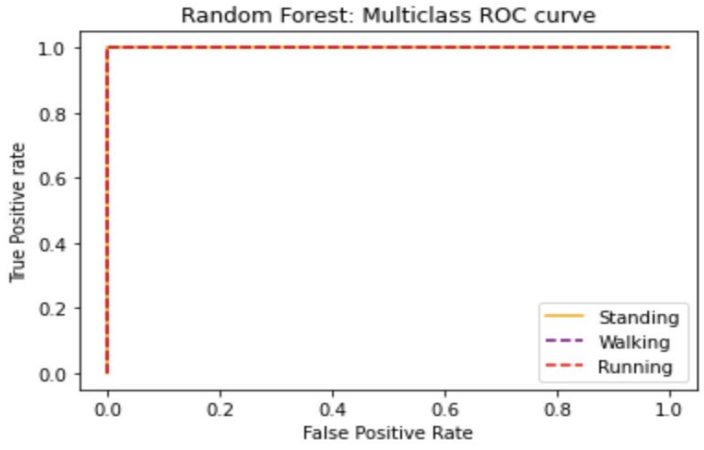
The classification report, Confusion matrix and ROC curves for all the classification models are given below. Though ROC works only for the binary classification, it was tweaked to work with multi classes because we had three classes in our mode. The results for the best model based on these parameters are explained in the next section(Best model).

1. Random Forest

```
Classifier :  Random Forest
Train Score:  1.000
Valid Score:  0.957

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.90      1.00      0.95         9
           2       1.00      0.86      0.92         7

    accuracy                           0.96        23
   macro avg       0.97      0.95      0.96        23
weighted avg       0.96      0.96      0.96        23
```

Confusion Matrix

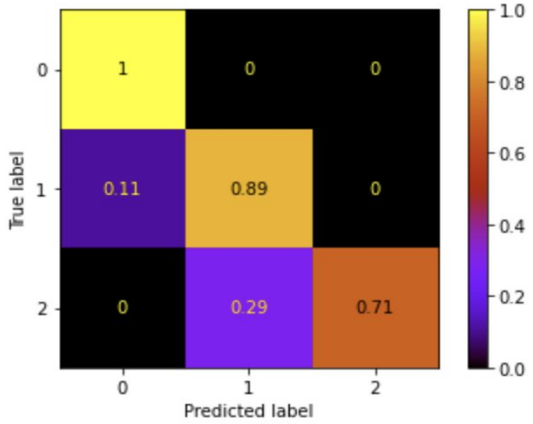


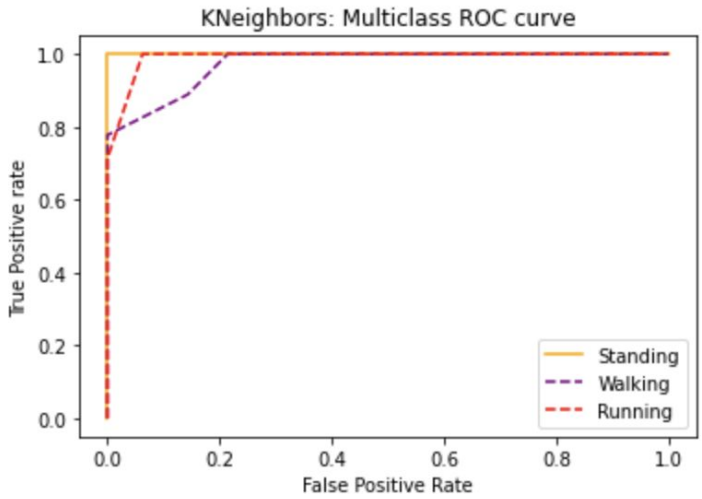


2. KNeighbors

```
Classifier :  KNeighbors
Train Score:  0.940
Valid Score:  0.870

Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.93         7
           1       0.80      0.89      0.84         9
           2       1.00      0.71      0.83         7

    accuracy                           0.87        23
   macro avg       0.89      0.87      0.87        23
weighted avg       0.88      0.87      0.87        23
```

Confusion Matrix

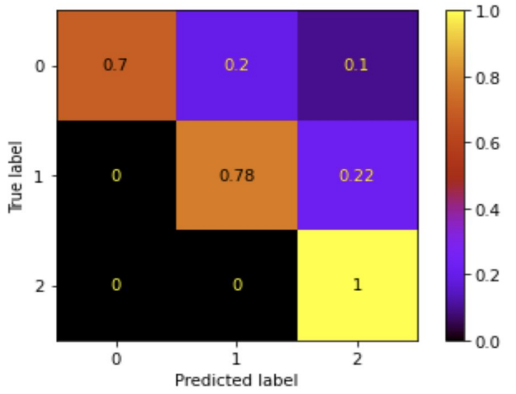


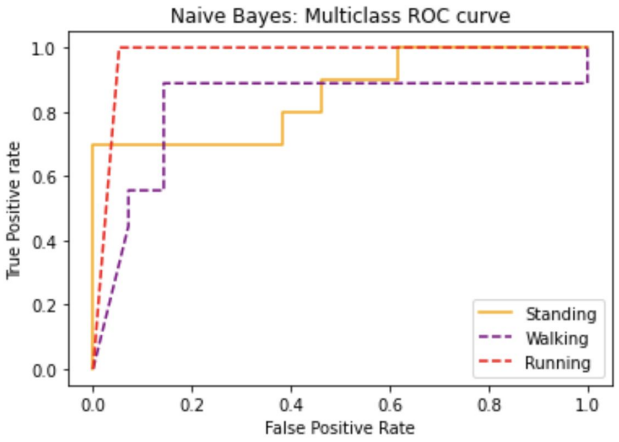


3. Naive Bayes (Gaussian)

```
Classifier :  Naive Bayes
Train Score:  0.955
Valid Score:  0.783

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.70      0.82        10
           1       0.78      0.78      0.78         9
           2       0.57      1.00      0.73         4

    accuracy                           0.78        23
   macro avg       0.78      0.83      0.78        23
weighted avg       0.84      0.78      0.79        23
```

Confusion Matrix

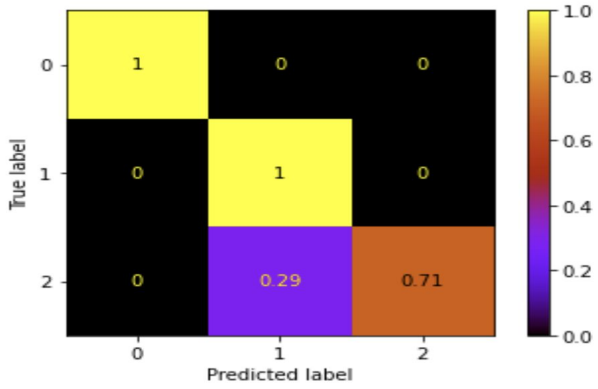


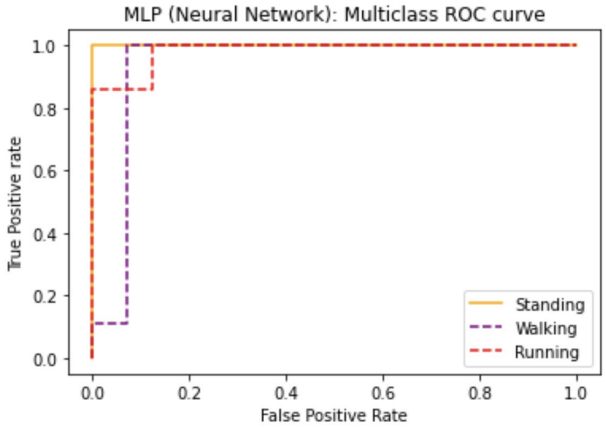


4. Neural Nets (MLPClassifier)

```
Classifier :  MLP (Neural Network)
Train Score:  1.000
Valid Score:  0.913

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.82      1.00      0.90         9
           2       1.00      0.71      0.83         7

    accuracy                           0.91        23
   macro avg       0.94      0.90      0.91        23
weighted avg       0.93      0.91      0.91        23
```

Confusion Matrix

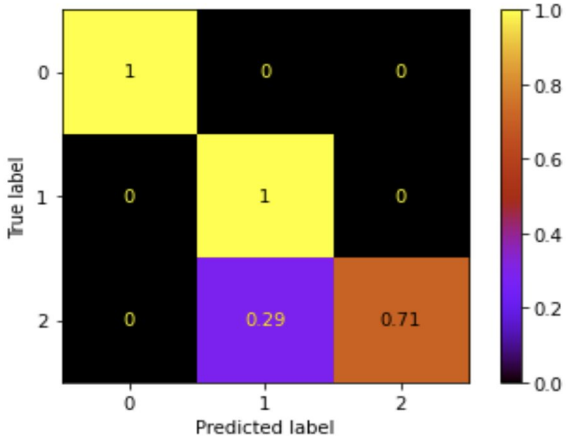


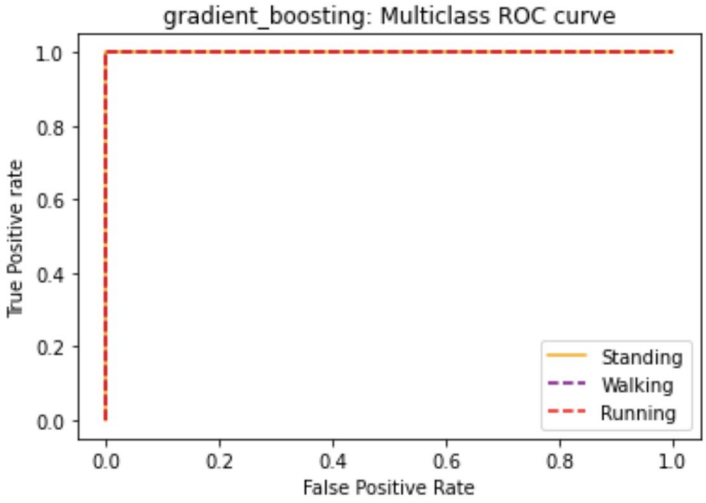


5. Gradient Boosting

```
Classifier :  gradient_boosting
Train Score:  1.000
Valid Score:  0.913

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.82      1.00      0.90         9
           2       1.00      0.71      0.83         7

    accuracy                           0.91        23
   macro avg       0.94      0.90      0.91        23
weighted avg       0.93      0.91      0.91        23
```

Confusion Matrix

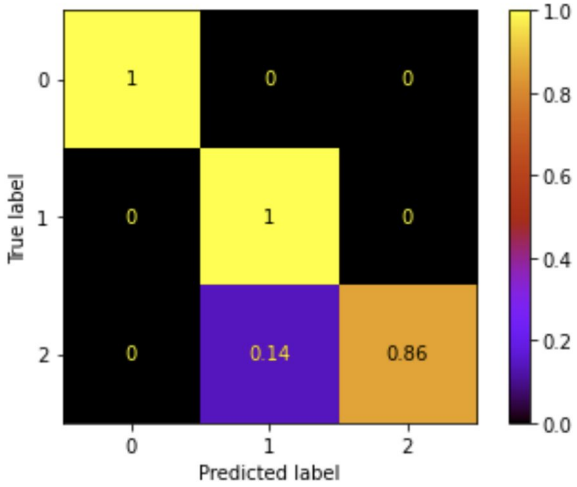


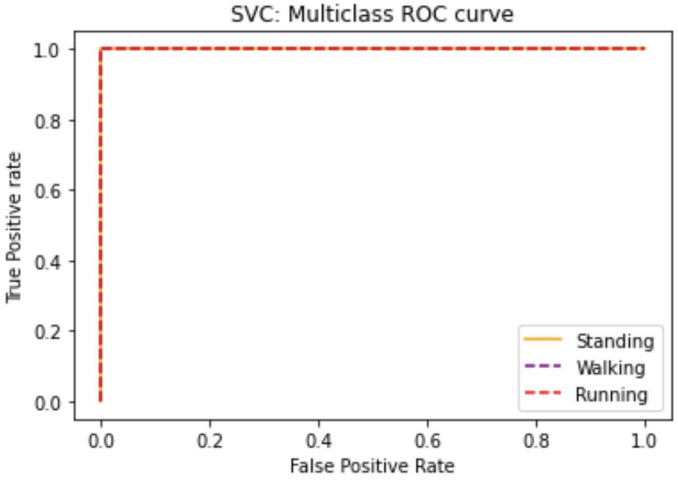


6. SVC

```
Classifier :  SVC
Train Score:  0.970
Valid Score:  0.957

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.90      1.00      0.95         9
           2       1.00      0.86      0.92         7

    accuracy                           0.96        23
   macro avg       0.97      0.95      0.96        23
weighted avg       0.96      0.96      0.96        23
```

Confusion Matrix

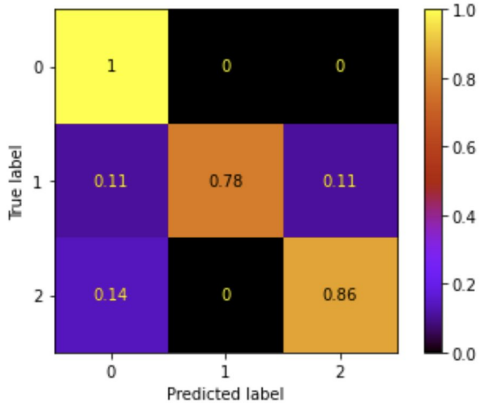


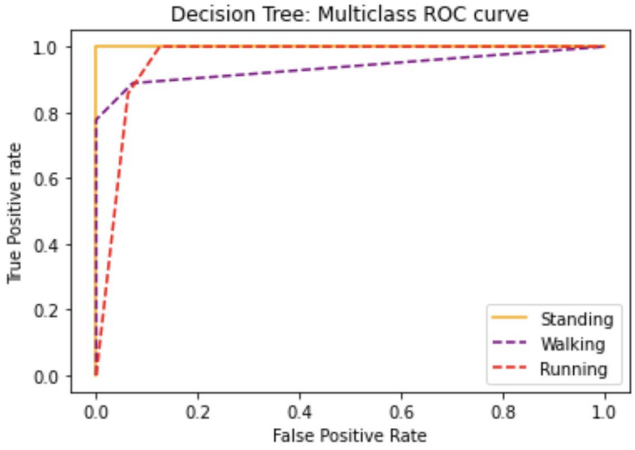


7. Decision Tree

```
Classifier :  Decision Tree
Train Score:  0.940
Valid Score:  0.870

Classification Report:
              precision    recall  f1-score   support

           0       0.78      1.00      0.88         7
           1       1.00      0.78      0.88         9
           2       0.86      0.86      0.86         7

    accuracy                           0.87        23
   macro avg       0.88      0.88      0.87        23
weighted avg       0.89      0.87      0.87        23
```

Confusion Matrix





3. **Best Model:**

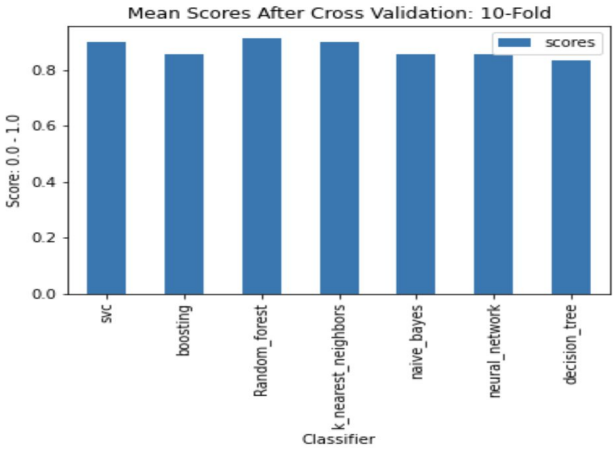
The random forest classifier seems to give the best results based Cross validation as well as the ROC,

classification and Confusion matrix.

Cross-validation: The mean validation score for 10 folds is approximately 91 percent which is higher than any of the other classifiers used (Shown by bar graph and table below).

K-fold Mean Scores:

	scores	classifier
0	0.900000	svc
1	0.855556	boosting
2	0.911111	Random_forest
3	0.900000	k_nearest_neighbors
4	0.855556	naive_bayes
5	0.855556	neural_network
6	0.833333	decision_tree

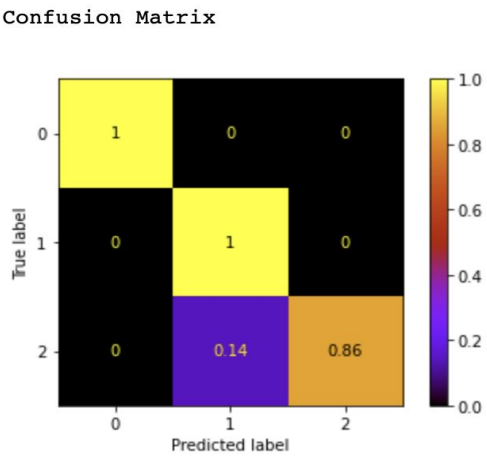


Classification report and Confusion matrix: The classification report gives a training score of 100 percent and validation score of 95.7 in case of just 1 split of data. This score was better than scores given by other models. Furthermore, it shows that the category 0 (standing) is giving almost perfect results whereas the category 1 (walking) shows a little misclassification as precision is 0.90 and recall is 1.00. If we look at the confusion matrix then only category 2 (running) shows a little misclassification whereas the category 0 and 1 are doing perfectly fine as shown in the figure below.

Classifier : Random Forest
Train Score: 1.000
Valid Score: 0.957

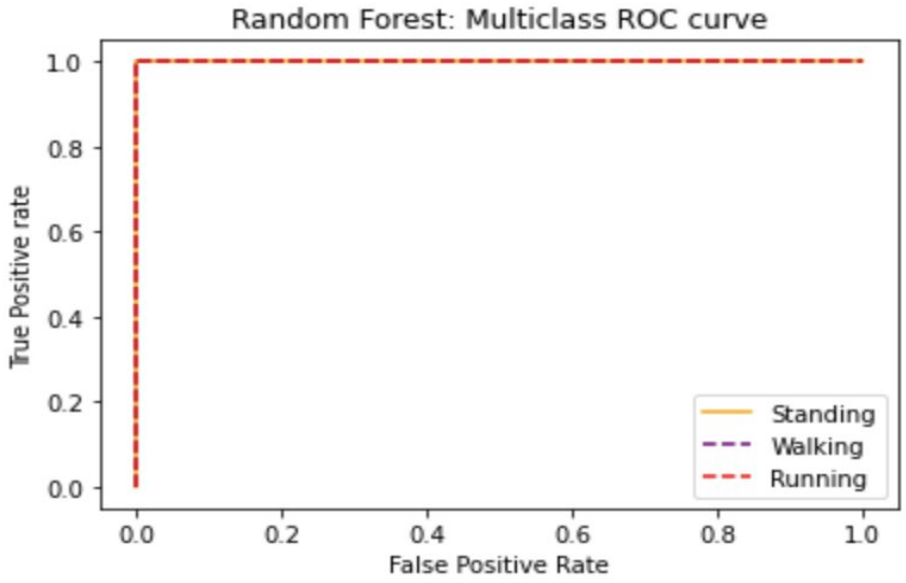
Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7
1	0.90	1.00	0.95	9
2	1.00	0.86	0.92	7
accuracy			0.96	23
macro avg	0.97	0.95	0.96	23
weighted avg	0.96	0.96	0.96	23



ROC curve: The ROC curve was tweaked to work with 3 classes though it is a binary classifier. ROC is

basically a plot between true positive rate and false positive rate. It tells how much the model is capable of distinguishing the classes. The ROC curve for the random forest shows no overlapping at all which shows that it is perfectly able to distinguish between positive class and negative class (shown in the figure below).



6. Interpreting & Presenting results

Prediction

All the models used in MachineLearning.ipynb were saved in "*.joblib*" format to use for testing. To test our modelling, we recorded 4 additional sets of data for each of standing, walking, and running and stored them in "***data/testdata/***". Out of the 12 files, any file can be fed through the terminal using a command like:

python3 main.py data/testdata/*walk3.csv*

This program will make a prediction using our best 3 models which have been saved in *".joblib"* format in "*models/*". The output from this program will be a list of predictions using each model and their probabilities. Following is a sample output:

```
Predictions for Dataset:
        0: Standing
        1: Walking
        2: Running

Classifier-wise Predictions
   Random Forest  MLP  SVC
0              1    1    1


Random Forest Probability
   Standing   Walking  Running
0  0.080645  0.919355      0.0


MLP Probability
   Standing   Walking       Running
0  0.000096  0.999904  1.137951e-15


SVC Probability
   Standing   Walking  Running
0  0.140293  0.799507   0.0602
```

Limitations

- It was our group's first time experience working on data analysis so we did not know how much data would be adequate for our analysis. We started working on the project thinking that we have enough data but later realized that machine learning models could have been trained in a better way if we had more time.
- We recorded the data using the accelerometer of the phone and after visualization of the data we realized that the measurements could have been much more accurate if we used a proper measuring device.
- In a few of the measurements the phone was attached to the ankle and while taking running and walking measurements it was very difficult to take time off the ankle and stop recording simultaneously. The waves for the running and walking were damped in the end. This could trick the model as if the measurement was taken while standing.

Future Prospects

- Our original idea was to determine if a person's activity readings point to a heart problem. We attempted to collect heart rate data using an Apple Watch but due to the inconsistent frequency of

the recorded data, we couldn't use it. For future, if there's an access to a device which can record both heart rate and data related to movement, it is possible to determine if a person has irregular heart rate by testing against recorded data for the same person and possibly determine if they need any immediate medical attention.

Accomplishment Statements

Arpit Kaur

- Collectected data for running, standing and walking.
- Worked on cleaning the data and applied a LOWESS filter to reduce noise.
- Transformed data to create engineered features out of raw data for statistics and machine learning.
- Worked on the overall structure, data acquisition and data cleaning part of the Final report.
- Worked on improving the readability of the code by adding comments and headings.
- Worked in analysis of machine learning models for final prediction.
- Worked on analyzing the ROC curves, Confusion matrix and classification report to evaluate the models.

Sharjeel Ahmad

- Collectected data for running, standing and walking.
- Worked on cleaning the data and applied butter worth filters to reduce noise.
- Made various plots for data visualization such as Histogram, Box plots etc.
- Performed Statistical tests on the engineered features to check statistical significance.
- Performed k-folds cross validation to evaluate the models.
- Worked on the Statistics and machine learning part of the report.

Srimalaya Ladha

- Collectected data for running, standing and walking.
- Worked on transforming the data to be used for machine learning models.
- Worked on creating all the classification models used for analysis.
- Worked on creating the ROC curves, Confusion matrix and classification report to evaluate the models.
- Worked on creating the final prediction application to test the project with the test data set.
- Worked on the machine learning part of the report to explain the best model.