Human Gait Activity Recognition

1. Abstract
2. Methods
   a. Phase 1: Preprocessing
   b. Phase 2: Feature Extraction
   c. Phase 3: Classification Techniques
3. Results and Conclusion
4. Challenges

1. **Abstract**

As we know Activity Recognition is useful in many fields as pervasive healthcare and surveillance, The recognition of human gait can be useful to identify the characteristics of the places or physical space such as walking, walking downstairs …etc.

In this Project a method of recognizing gait activities using acceleration data obtained from a smartphone, the acceleration signal were segmented based on the strides (Peaks and Valley) then Feature vector of the segmented signals was extracted, Which was used to train <u>five Classifier Using:</u>

-Support Vector Machine

- Decision Tree

- Random Forest

-Naive Bayes

- K-Nearest neighbors

<u>Our Dataset:</u>

It was collected from data consist of six Activities [jogging ,stand, walking, stair up, stair down, sit]

Where each activity has its own accelerometer data and time stamps

2. **<u>Method</u>**

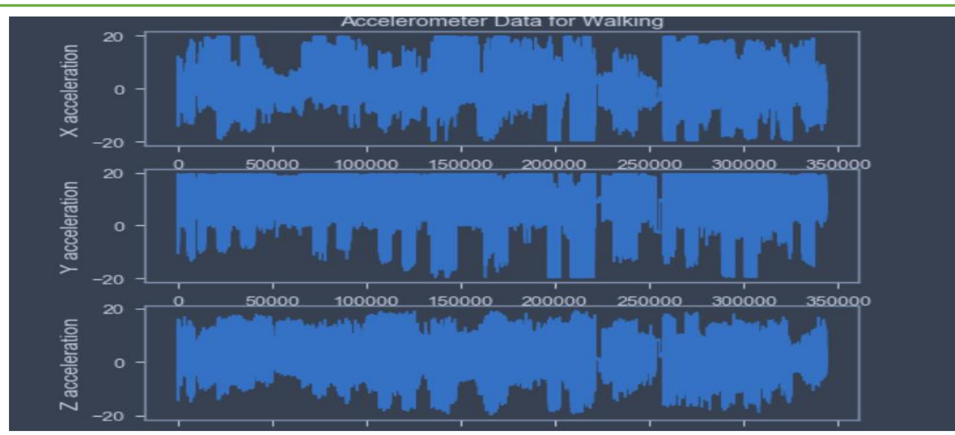
Our Method is Classified into 3 Phases

a. <u>Phase 1: Preprocessing</u>

i. Load Data

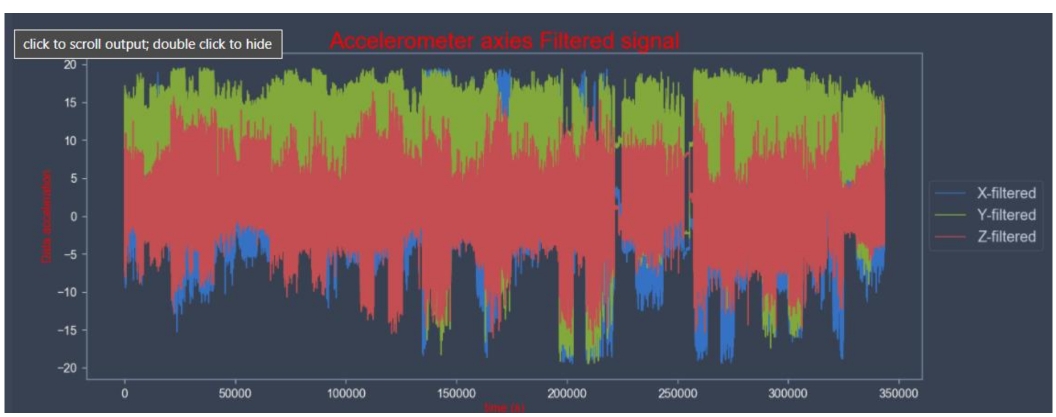
First, we load our data, then we find that it consists of 5 columns " time Elapsed, x, y, z, and label " and it consists of 343416 row

We plot Axes of Accelerometer Data:



ii. Appling Low_ pass Filter

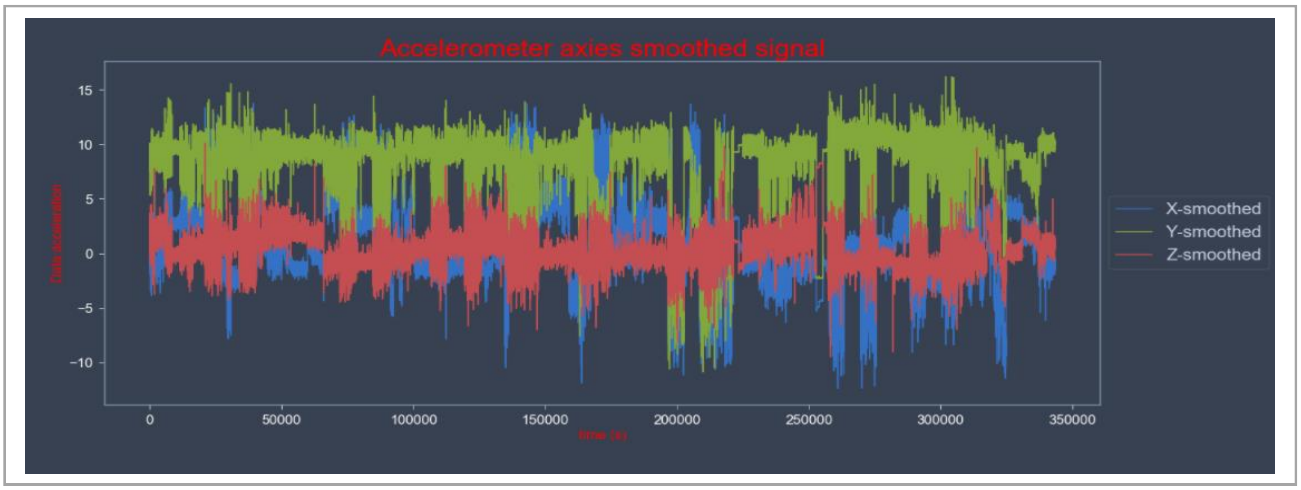
A low-pass filter is applied to accX, accY, accZ, we use the coefficient a which filter the highest frequencies with a **<u>value of 0.5</u>** the plot after filtration was :



iii. Applying smoothing

By using gaussian filter1d from SciPy library with sigma value = 4 we smoothed the

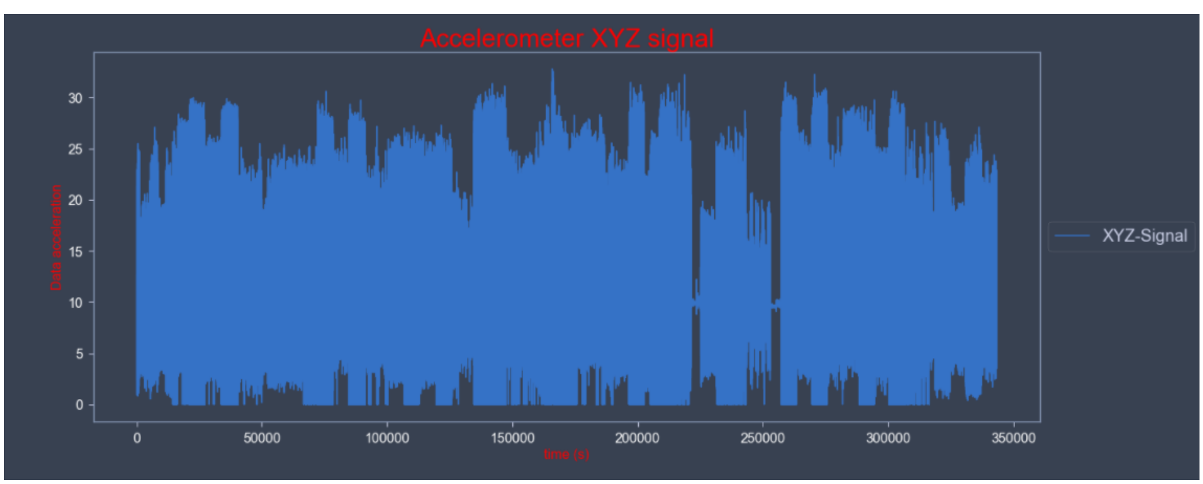
accelerometer axes data and the plot :

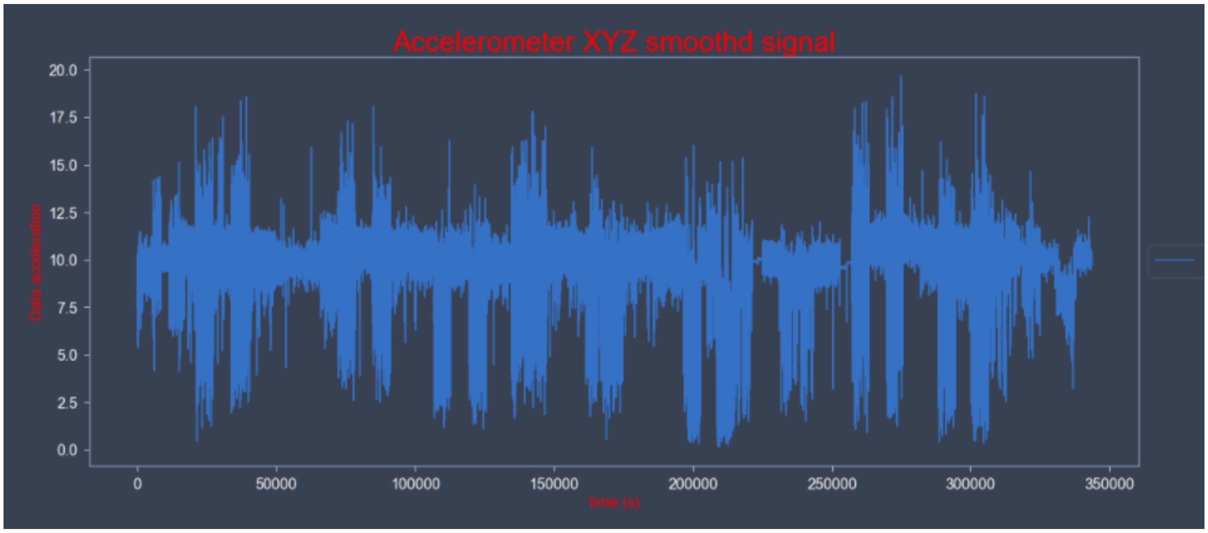


iv. Applying XYZ

Now we try to find the magnitude of the three axes and w make this on smoothed and filtered x, y, z acceleration data by applying $\sqrt{x^2 + y^2 + z^2}$

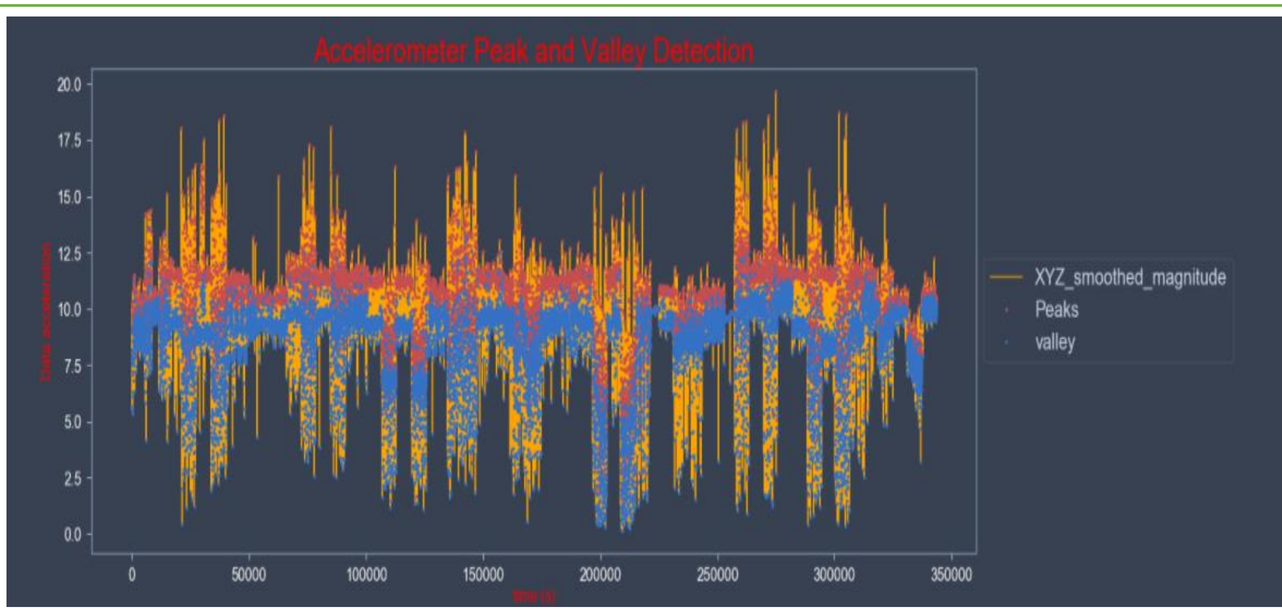
Plot for Data after applying:





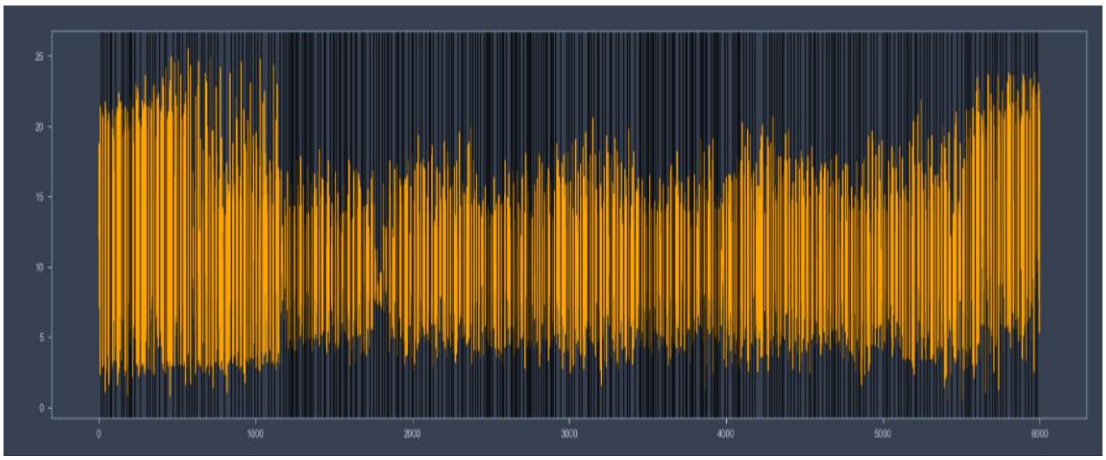
v. Detection of Peaks and Valleys

We use find_peaks function from SciPy library we use it also for detecting valleys by multiplying XYZ_smoothed values with -1 and the peaks



vi. Segmentation

Applying segmentation depends on valleys of dataset XYZ_Smoothed then we apply this segmentation on XYZ_magnitude:



b. Phase 2: Feature extraction

i. Peak Height

To find the height of the Peak we started with detecting the maximum point and the minimum point in the peak and then we subtracted the minimum from the maximum.

ii. Mean

To find the mean of the wave we detected the period of the wave and used all the points in the wave to calculate the mean.

iii. Peak width

To find the width of the peak we used time steps by subtracting the time of the end of the wave from the beginning of the wave.

iv. Standard Division

To calculate the standard deviation we did the same as in the mean calculation phase.

v. Data Frame for feature

After we extract features we put this feature in the data frame (df_feature) then we labeled each features row according to the actual label
for example: as we know them from time 0 to 99 there is an activity done which is walking, the first 12 valley points are from 0 to 90 so they are labeled as Walking and so on.
We found feature data consists of 151 rows and 5 columns.

c. Phase 3: Classification

i. Train_Test_Split

We split the Feature data set to X And Y where
X is (acc_mean, acc_Std, peak_height, peak_width)
Y labels.
Then we use function train test split from the scikit-learn library and split data with 0.1 for test and random state = 49.

ii. Preprocessing for Features data

we use here preprocessing label encoder to convert Y to label and Standard Scaler to make feature scaling to X data.

iii. Algorithms Used

- Support Vector machine:

  We use Kernel 'rbf' , random state =0 and fit with train data and predict with test

- K-Nearest Neighbors:

  We use n_neighbor =3 metric= 'minkowski' p=3

- Random forest:

  n_estimators = 17, criterion = 'entropy', random_state = 0

- Decision Tree: max_depth = 5

- Naive Bayes:

Confusion Matrix & Cross Validation:

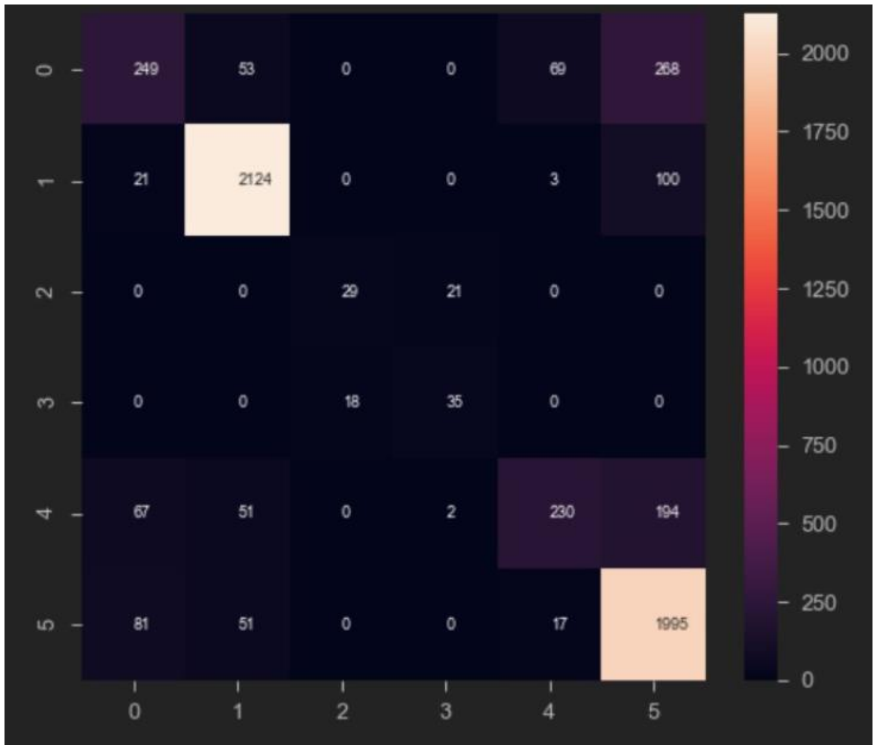
For each algorithm, we have used Cross-validation in all algorithms to training and testing data to estimate the skill of machine learning models.

- In using Cross-validation for each data using a certain algorithm there is a resampling procedure used to evaluate machine learning models on a limited data sample and generates a score for each sample.

Then we have used confusion-matrix and applied it to all the Y-test and Y-prediction data to have a better idea of what the classification model is getting right and what types of errors it is making.

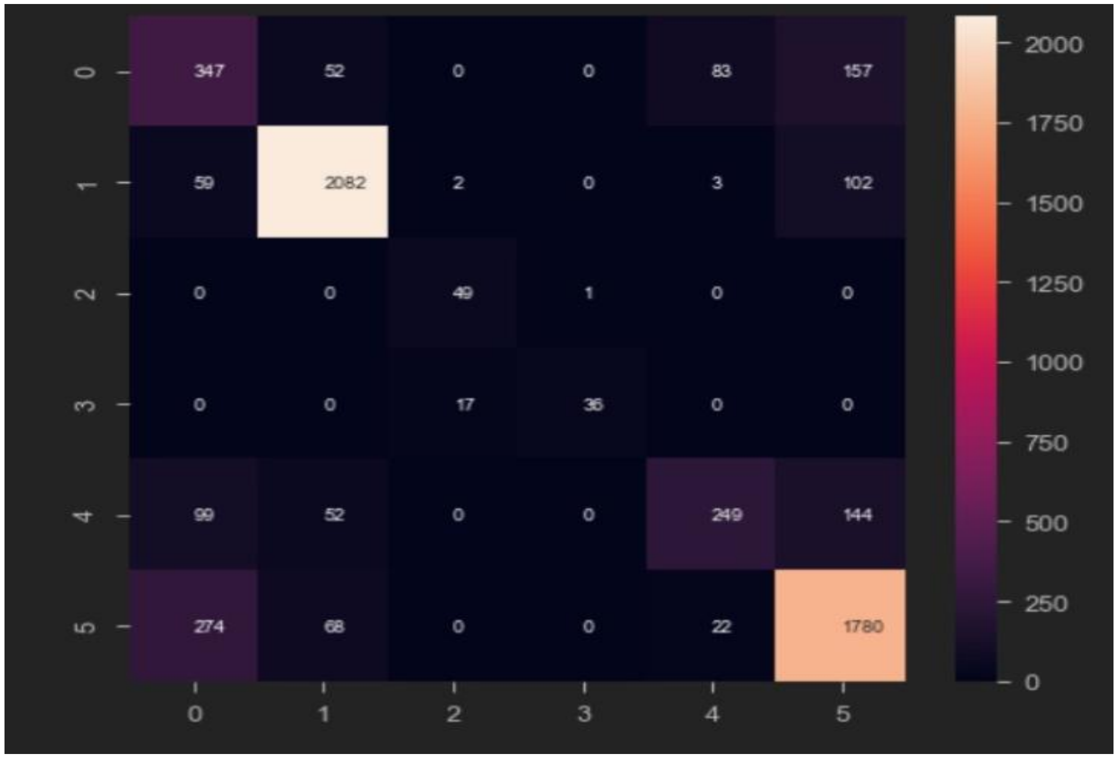
Confusion Matrix:

For SVC:



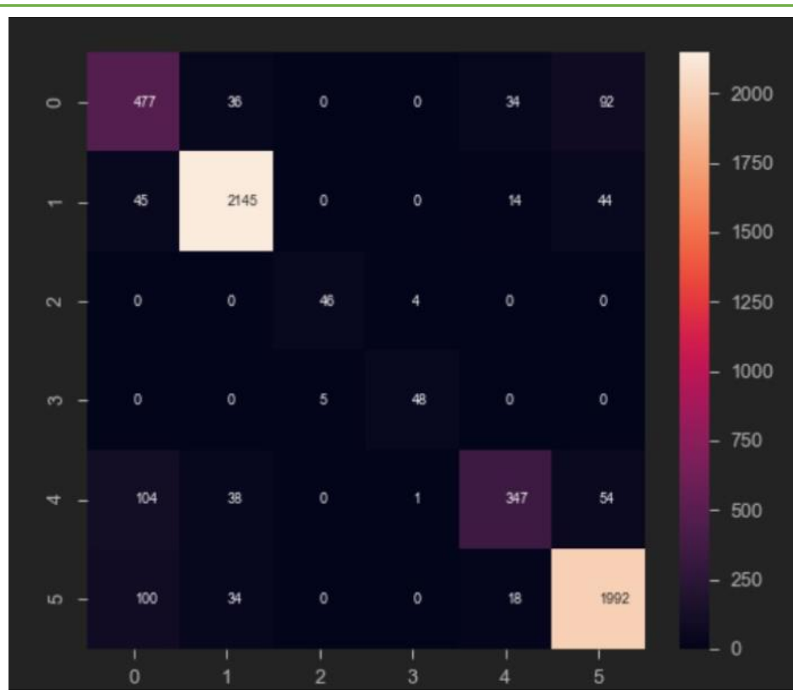

```
Cross validation for train data SVC is :  [0.80551593 0.7787288  0.66613824]
Cross validation for test data SVC is  :  [0.81035394 0.79609086 0.82082452]
```

For DT:



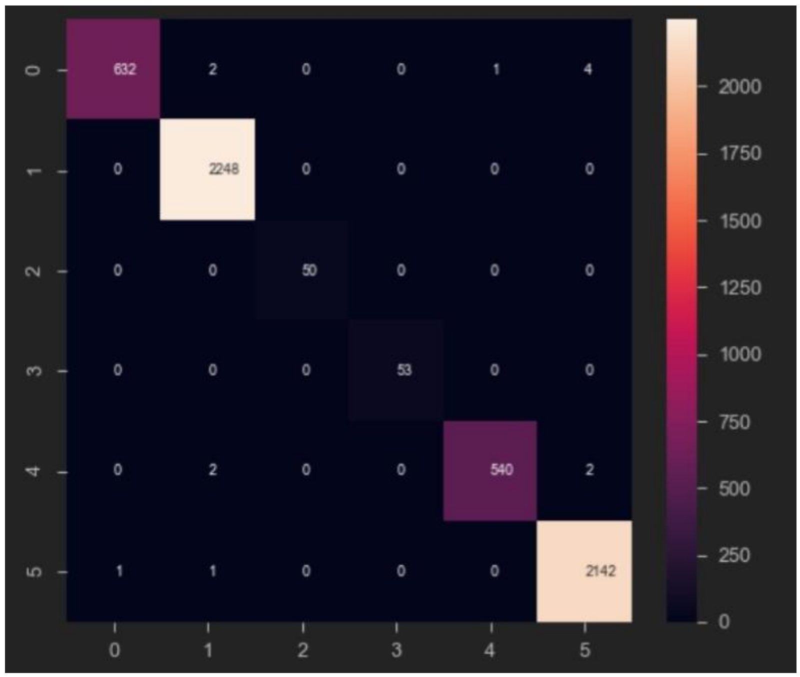

```
Cross validation for train data DT is :  [0.78094785 0.76921858 0.63649334]
Cross validation for test data DT is :  [0.78816693 0.7691495  0.79968288]
```

For KNN:



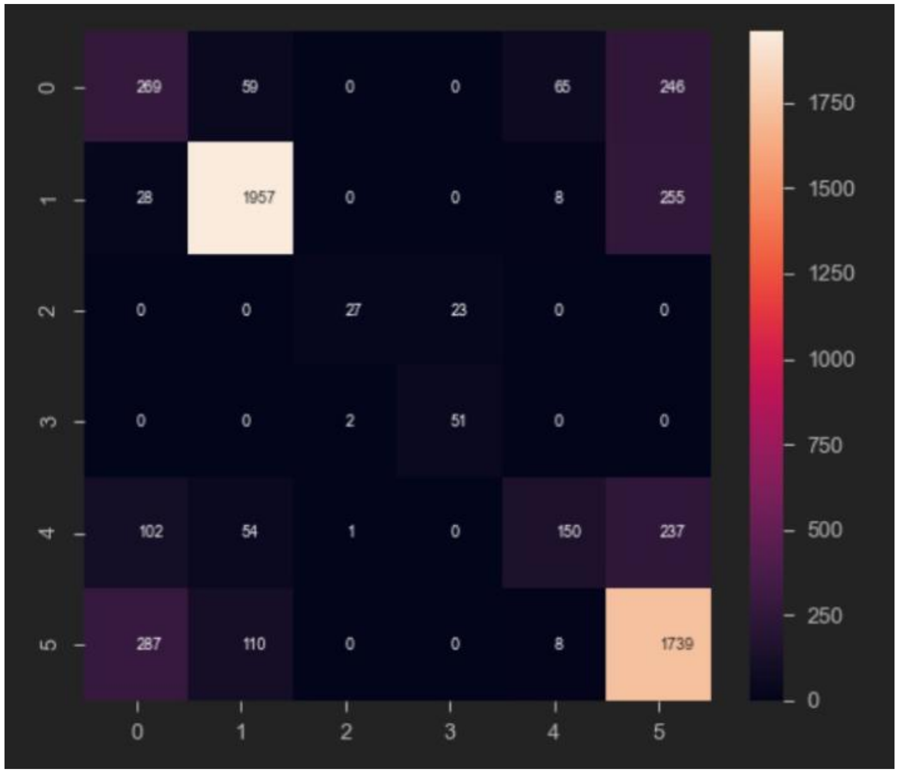
```
Cross validation for train data KNN is :  [0.74433349 0.75701379 0.66169943]
Cross validation for test data KNN is :  [0.80559958 0.79080824 0.80232558]
```

For RF:




```
Cross validation for train data RF is :  [0.77650975 0.76921858 0.67866202]
Cross validation for test data RF is :  [0.8219757  0.79926043 0.82135307]
```

For NB:




```
Cross validation for train data NB is :  [0.76287843 0.75273419 0.6236525 ]
Cross validation for test data NB is :  [0.74484945 0.73164289 0.74735729]
```

3. Results and conclusions:

To Calculate accuracy we use the **accuracy score** function from metrics in sklearn

Algorithm	Accuracy
SVC	0.82
Decision Tree	0.8
Random forest	0.99
KNN	0.89
Naive Bayes	0.74

When we increase dataset size accuracy increase.

Test model with new data:

Algorithm	Accuracy
SVC	0.33
Decision Tree	0.31
Random forest	0.32
KNN	0.29
Naive Bayes	0.34

```
Accuracy of SVC          ---------> 0.33
Accuracy of Desicion Tree---------> 0.31
Accuracy of Random forest---------> 0.32
Accuracy of KNN          ---------> 0.29
Accuracy of Navie Bayes  ---------> 0.34
```

4. Challenges:

We Faced a lot of challenges as the paper of Recognition of Gait Activities Using Acceleration Data from A Smartphone and A Wearable Device. Has many functions with no identification for it like Power( ), Forward Direction, segmenting, power for feature extraction, and we couldn’t produce the classification result as TPR and TNR because it’s not mentioned in the paper. Also dataset we didn’t find a perfect dataset, So we started collecting the dataset manually from different accelerometers read for activity and it was so small.