CS7CS4: MACHINE LEARNING

Group 80

SHRITESH JAMULKAR
VIPUL GHARE
ADITYA RADHAKRISHNAN

Introduction

Over here, our group will be aiming to predict the fall of an old person during his/her day based on the reading of their activity. Some injuries may be fatal if necessary help doesn't arrive on time. Thus, lives can be saved by tracking these falls with the use of technology. If the readings indicate their fall, a feature to notify hospitals or close relatives would come in handy in avoiding this mishap.

We aim to solve this problem with the help of intelligent sensors and Machine Learning Techniques. We have considered gyroscope and accelerometer readings(x,y,z) as input features for our model to predict the activity of a person.

A mobile cross-platform *React-native* application is built for collecting data from mobile sensors using *expo* libraries. Each data point is contained every 0.5 seconds. An *Accelerometer* and *Gyroscope* from mobile sensors are used to collect data. *Figure 1* shows the methodology followed to gather mobile sensor data.

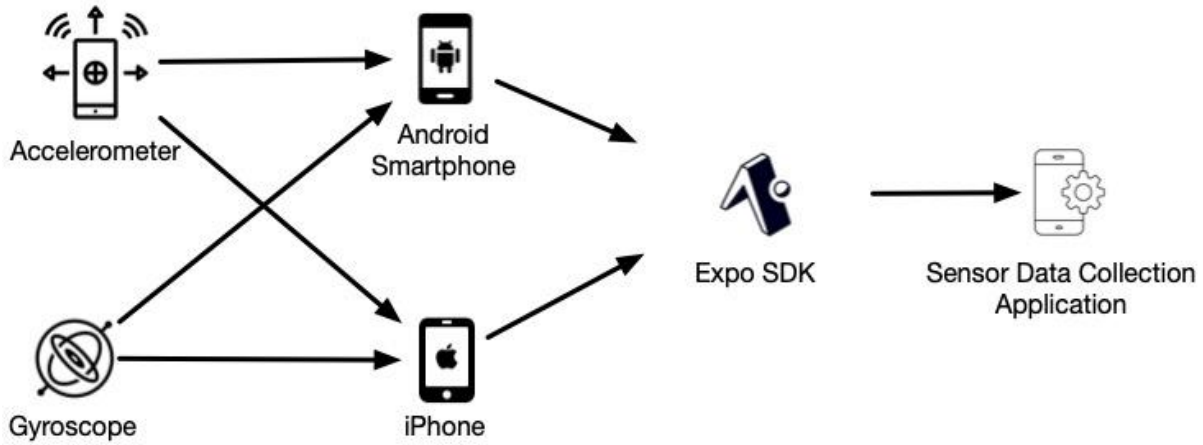


Figure 1: Gather Mobile Sensor data using Expo SDK

We tried to predict the fall of a person by using 3 different classification techniques, namely *Long Short Term Memory* (LSTM), *K-Nearest-Neighbours*(KNN), and *Random Forest Classifier*. The performance of all these models is compared with the baseline model which always predicts the most frequent class in the dataset.

Dataset & Features

The dataset consists of a total of 7 features ie. readings from an accelerometer (accx, accy, accz) and from a gyroscope (gyrx, gyry, gyrz) which were taken simultaneously along with their corresponding timestamps that would record the event. As the application is installed on our mobile

phones, we tried to collect the readings of the accelerometer and gyroscope by holding the mobile in our hands while performing different activities. Activities such as ‘walking’, ‘jogging’, ‘upstairs’, ‘downstairs’, ‘sitting’, and ‘falling’ were considered for the preparation of our dataset. The sample videos on how the data was collected have been added to the GitHub repository.
Our mobile application allows us to export the readings of the accelerometer and gyroscope in a CSV format. We took readings for each of the activities(for e.g. walking, sitting etc), based on their accelerometer and gyroscope readings along with a fixed duration for each activity, for e.g. in the ‘walking’ activity, we collect and merge the accelerometer & gyroscope readings with their corresponding timestamps. We repeated this activity ‘5’ times to collect an adequate dataset. We then concatenate these 5 samples and compile it into a CSV file.

We repeated this process for the remaining 5 activities as well, and finally concatenated them, which produced our main CSV file *‘MainDataset’*. The features of the MainDataset were ‘accx’,‘accy’,‘accz’,‘gyrx’,‘gyry’,‘gyrz’ and ‘Time’. The below figure depicts the data points for activity collected.

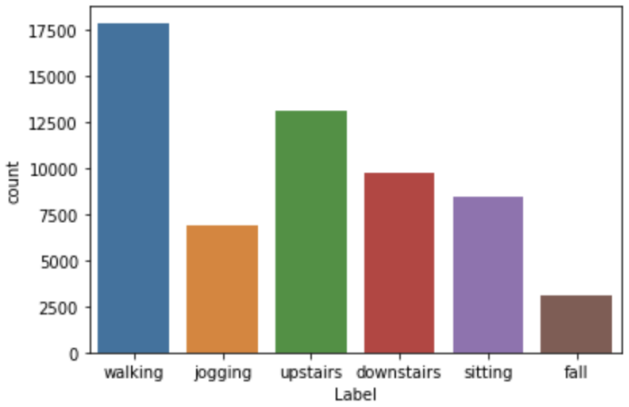
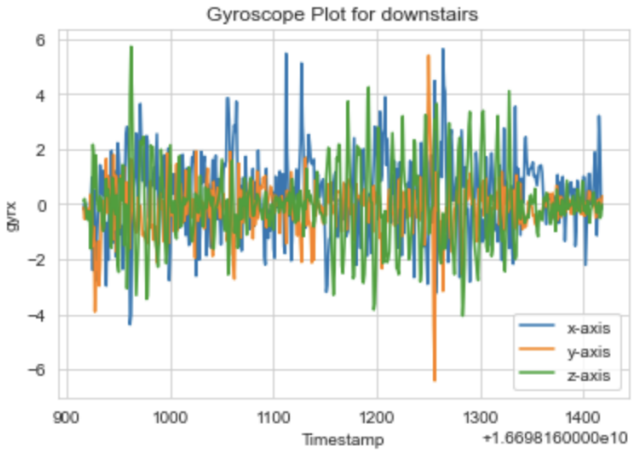


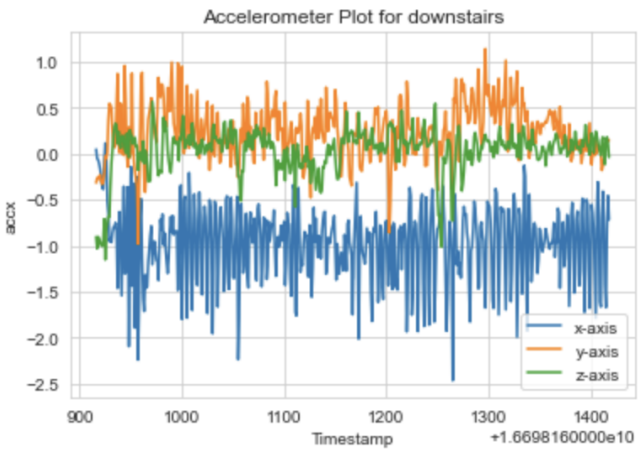
Figure 2: Distribution of data w.r.t. Activites.

As we can see ‘walking’ class has the majority of the data points, which are around 17,500. While the ‘fall’ class contains the least data points.

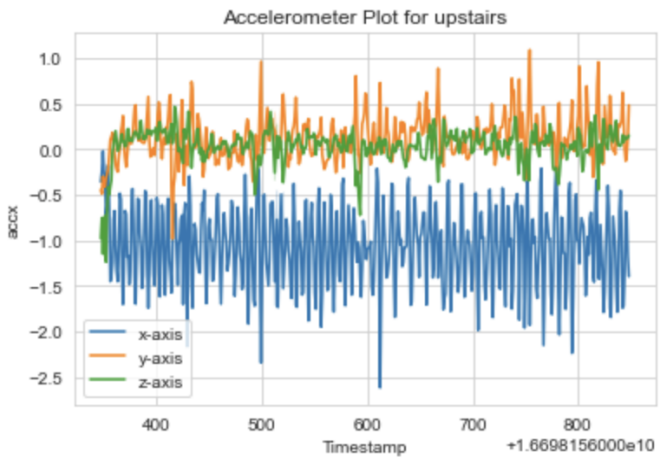
Below plot depicts the accelerometer and gyroscope data with respect to the timestamp.

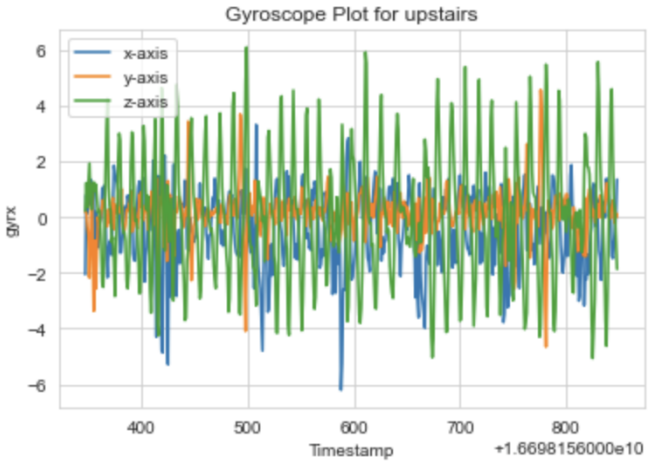
Data for Downstairs :



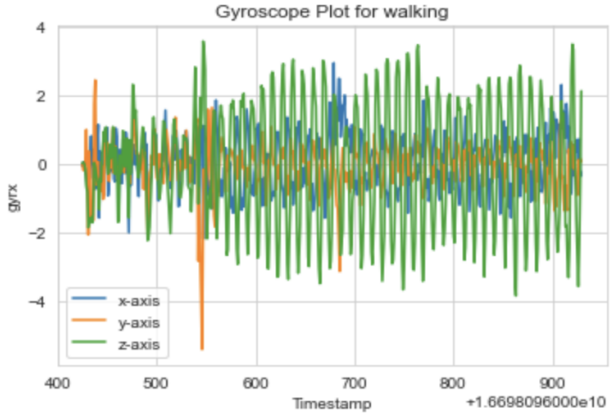


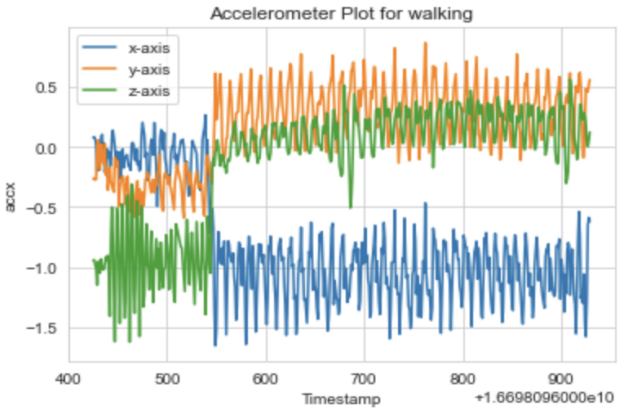
Data for Upstairs :



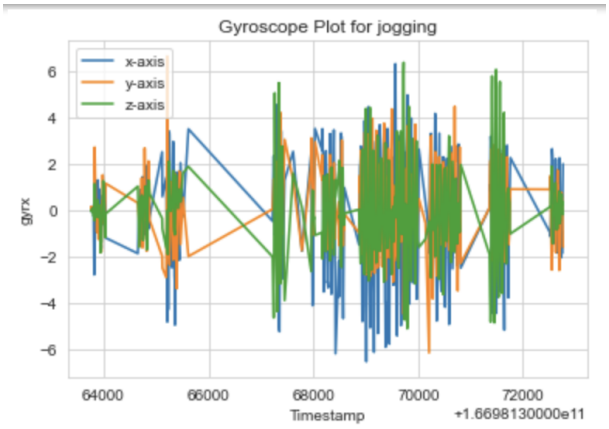


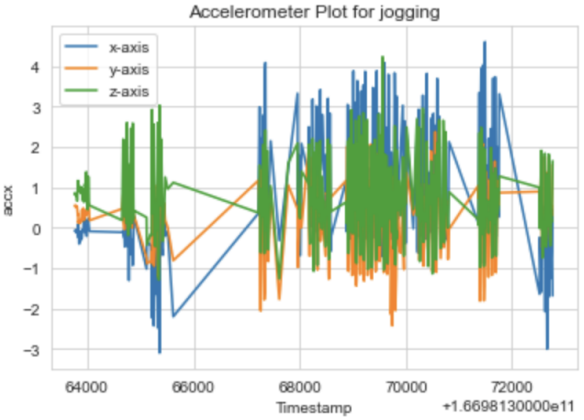
Data for Walking :



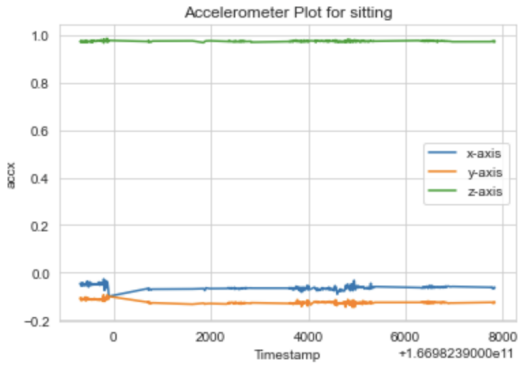


Data for Jogging :

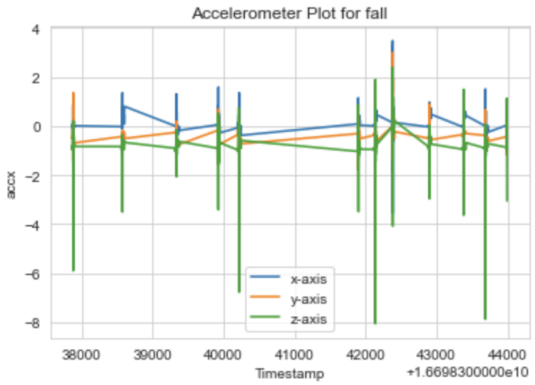


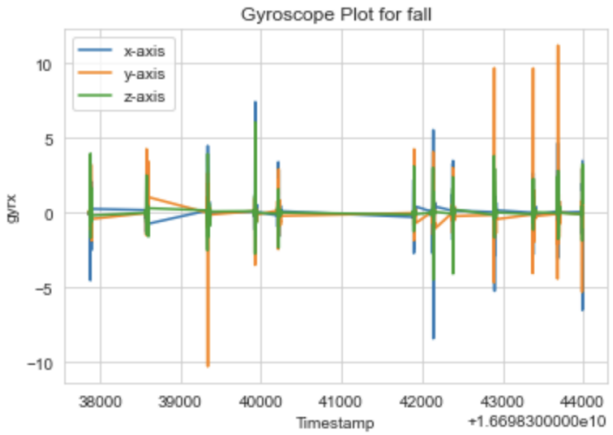


Data for Sitting :



Data for Falling :





Methods:

Long Short-Term Memory (LSTM)

According to Wikipedia Long Short Term Memory is a type of Recurrent neural network capable of learning long-term patterns in data. It is primarily used in fields like sequence predictions like a stock market prediction. A typical LSTM layer consists of a cell state, LSTM an input gate, an output gate and a forget gate. The cell state helps information to be passed through the gates and selectively decides which information to keep. Since our problem statement requires sequential processing of data hence, we are using the LSTM model to predict different labels as described in the dataset.

K-Nearest Neighbors(KNN)

The '*K - Nearest Neighbours Algorithm'* approach is based on checking the proximity between an unknown data point and its neighbours and classifying it further. This is done by calculating the distance between them, and telling us which class of neighbours the unknown data point hinges towards. Thus, getting classified, making this a distance-centric approach. This lazy algorithm saves us quite some time as it doesn't need a training period. However, it takes time w.r.t to computation as it makes use of memory and takes time in storing datasets. In order to determine an ideal 'K' value, we will perform cross-validation on the model. Overall this is a simple approach to implement.

Random Forest (RF)

Random Forest algorithm uses an ensemble technique along with multiple decision trees to make predictions. As the model is made by considering many predictions, overfitting can be avoided to some extent. The random forest can tell us the importance of one feature in the dataset to make predictions. Random forest performs well on large datasets where many numbers of decision trees are sampled from the original dataset to make combined predictions. We can tune our model using cross-validation on the number of trees, depth, minimum sample per split, etc.

Experiments/Results/Discussion:

Long Short-Term Model (LSTM)

To predict the person's activity we take accelerometer and gyroscope readings as our input features and with the step size of 10 and 6 sets of features i.e. accx,accy,accz,gyrx,gyry,gyrz.

Model Architecture

The selection of the correct set of layers is very important in the problem statement we are trying to solve hence, we have used the sequential areas model to implement our Nural Network. We have implemented different LSTM models by playing around with different numbers of LSTM layers and fully connected layers and finally we decided to have 4 layers out of which the first layer is the LSTM layer with 128 units and the second layer is a dropout layer with a frequency of 0.5. The next layer is the fully connected layer with the Relu activation function to downsample the result of the previous layer and finally, the output layer with the Softmax activation function which is used to predict the number of classes. Totally, the model has 77,766 trainable parameters.

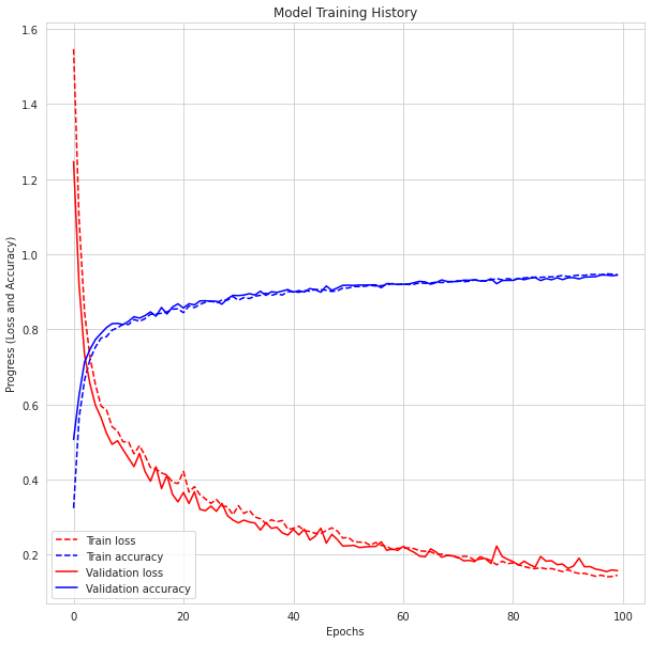


Figure 3.1 : LSTM Model Training History

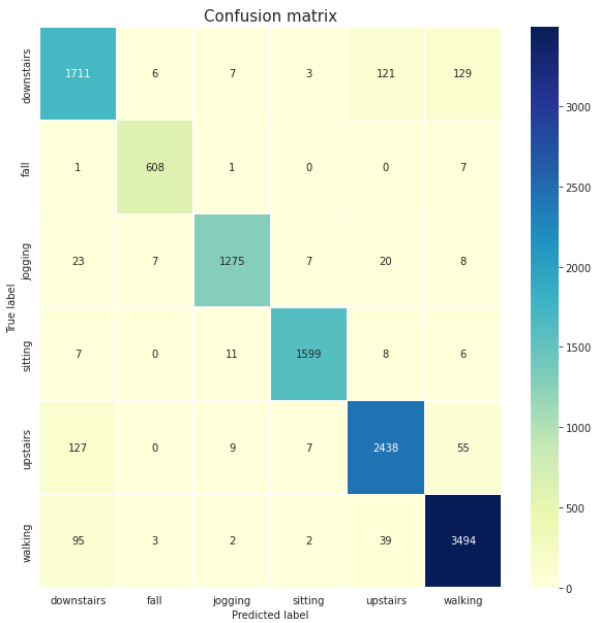


Figure 3.2 : LSTM Model Confusion Matrix

Model Evaluation

We have trained the model on Google Colab with GPU support. The training took around 42 seconds for 100 epochs with a batch size of 1024. Figure 3.1 shows the history of model learning with increasing accuracy and decreasing loss. The LSTM model achieved an accuracy of 93%. Figure 3.2 shows the confusion matrix of the LSTM model.

Hyperparameter tuning

We have tried to tune the Batch size and learning rate and epochs as a part of hyperparameter tuning. The model is yielding highest accuracy when the batch size is 1024 on 100 epochs with a default learning rate of 0.01. On increasing epochs, above 100 the model starts to overfit the data, and with 50 epochs the model is yielding less accuracy.

K-Nearest Neighbors Model(KNN)

To predict the person's activity we take accelerometer and gyroscope readings as our input features i.e. *accx, accy, accz, gyrx, gyry, gyrz*. We will train our model over a range of K values by having our training data split up into train and test data, taking an ideal split ratio of 80/20 by convention. Then, we perform a cross-validation to select a suitable 'K' value.

Hyperparameter tuning

For the cross-validation we take the K values in the range of 2,510,15,25,50,100. With the help of an error bar to catch increasing K values against F1 scores (Fig 4.1). From this plot, we infer that the value 'K = 10' helps our model in achieving the highest F1 test score in the given ranges. Hence, we use the K value '10' as the neighbour candidate to train the model. A ROC curve was plotted based on our prediction(K = 10). (Figure 4.2)

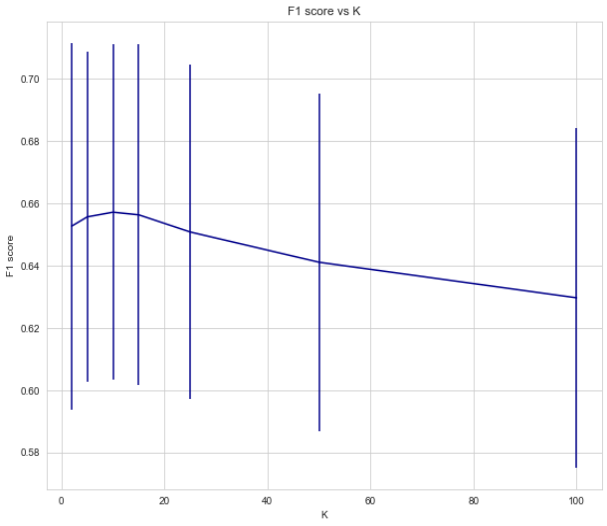


Figure 4.1

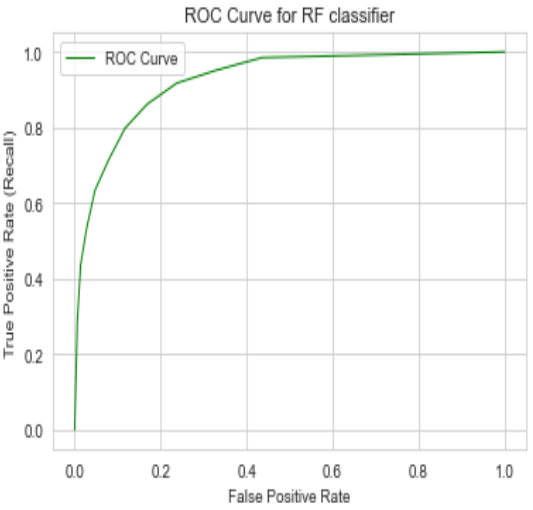


Figure 4.2

Results

Our K Nearest Neighbours model achieves an accuracy of 74.20%. with a K value of '10'.
The confusion matrix is depicted below *Figure 4.3* .

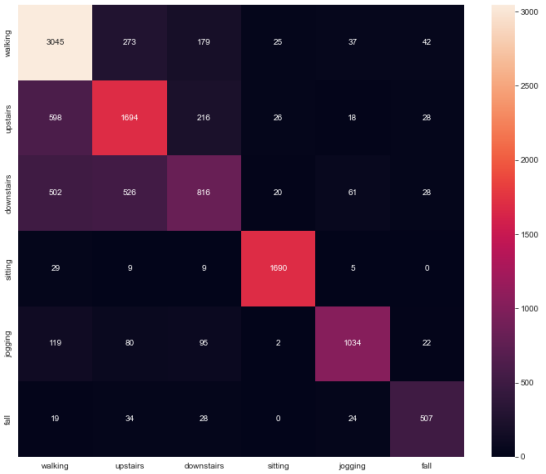


Figure 4.3 : Confusion matrix: KNN

Random Forest Model

To predict the activity of the person, we have trained our model using the Random Forest algorithm. The same input features (x,y,z) of the accelerometer and (x,y,z) of the gyroscope we used to predict the activity. To train our model we first divided our dataset into training and testing. The model is built on training data and to check the accuracy of the trained model, testing data is been used.

Hyperparameter Tuning

Cross-validation technique was used to tune our model. We performed the tuning of the “number of estimators” as it is one of the major parameters in a random forest. We selected a varied value of estimators such as [1,10,25,50,75,100,150,200]. An error bar plot was against each value of n_estimators vs F1 score. (Figure 5.1). From this we came to know that n=150 has the highest F1 value, so we created our final model by considering n_estimators as 150 and then made our predictions. The ROC curve was plotted based on our prediction by the model(ie n_estimators=150). (Figure 5.2)

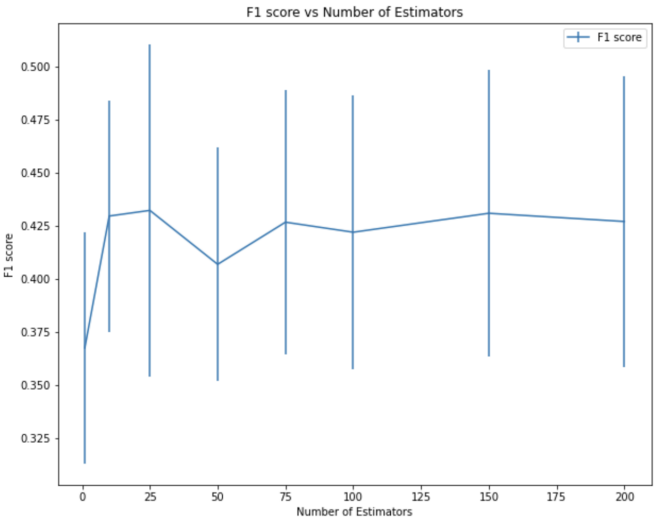


Figure 5.1

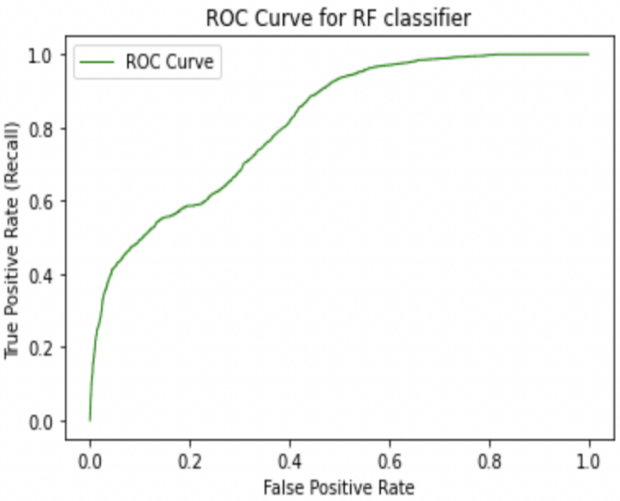


Figure 5.2

Results

Our random forest classification model with 150 n_estimators was able to achieve an accuracy of 58.09%. The confusion matrix is depicted below(Figure 5.3).

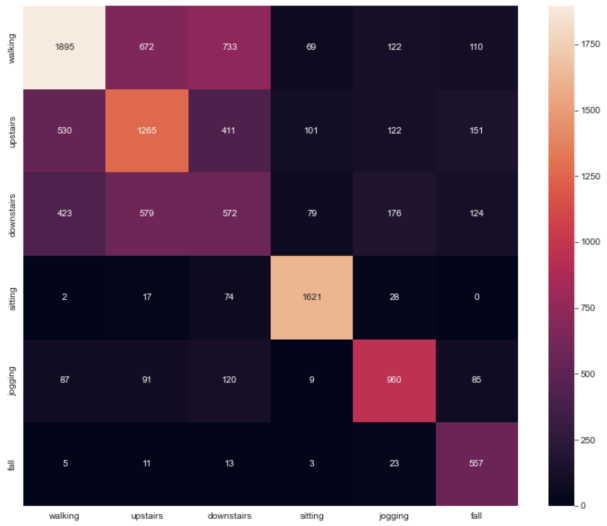


Figure 5.3 : Confusion Matrix for Random Forest Algorithm

Comparison with Baseline Model

We have selected the baseline model as the Baseline Classifier which always predicts the most frequent class in the training data. From section 2 we came to know the distribution of the classes in the dataset. The accuracy of our baseline model is about 30.41%. The following is the confusion matrix.(Figure 6)

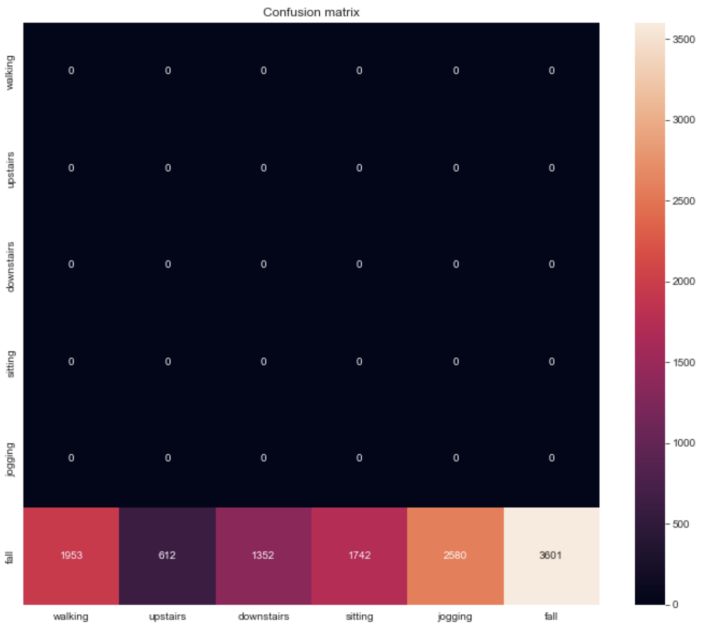


Figure 6 : Confusion matrix for Baseline Classifier(most Frequent)

Summary

We successfully trained 3 models namely Random Forest Classifier, K-Nearest-Neighbours, and Long Short Term Memory(LSTM) for the accelerometer and gyroscope dataset that we collected using a mobile application. We tuned the random forest classifier, the KNN model, by using a cross-validation method where we found the best value of the number of trees, and the number of nearest neighbours. However, the LSTM model stood out to be the most reliable model, giving an accuracy of almost 94%. Taking into account the quantity and quality of data available to train, the LSTM made very good predictions. The below table illustrates the accuracy of these models with respect to the baseline model.

Metrics	**LSTM**	**KNN**	**Random Forest**	**Baseline**
Accuracy	93.99%	74.20%	58.09%	30.41%

Contributions

The work of this project was done purely based on group work. We had discussions/meetings related to the group project topic to be selected. We used to take a quick daily update on the assigned task to keep the track of work.

Vipul Ghare
– Data Collection (ie performed activities such as sitting, walking, jogging, upstairs, and downstairs fall).
– Data Preprocessing/Data cleanup (wrote the code for data preprocessing ie Merging, and concatenation of numerous CSV files to make the main dataset).
– Performed Experiments with Random Forest.
– Creation of Baseline Model.
– Comparison of all the models .
- Worked on the report.

Aditya
- Data Collection (ie performed activities such as sitting, walking, jogging, upstairs, and downstairs fall).
- Worked on K Nearest Neighbour Algorithm.
- Performed Experiments on the KNN model and evaluated the model against the baseline.
- Worked on the report.

Shritesh
- Developed the mobile application for collecting sensor data.
- Designed the LSTM model and fine-tuned the model to improve accuracy and evaluated the model against the Baseline model.
- Worked on the report.

References & Links

- The below is the github link that contains the raw data, python code of data preprocessing, videos of how data was collected, all the csv files, Main csv file, and python code for execution of all the models. https://github.com/vipulghare/CS7CS4_MAchine_Learning_Group80.
- Expo is an open-source platform for making universal native apps for Android, iOS, and the web with JavaScript and React. https://expo.dev
- React Native is an open-source UI software framework created by Meta Platforms, Inc. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web, Windows, and UWP by enabling developers to use the React framework along with native platform capabilities. https://reactnative.dev/
- Long short-term memory. (2022, November 27). In *Wikipedia*. https://en.wikipedia.org/wiki/Long_short-term_memory