Human Activity Recognition Using Smartphones

Introduction :

The human ability to recognize another person's activities is one of the main subjects of study of the scientific areas of computer vision and machine learning. Because of this research, many applications, including video surveillance systems, human-computer interaction, and robotics for human behavior characterization, require a multiple activity recognition system.

Problem Statement :

The problem in question is a multi-class Classification problem. The input data consists 3-axial linear acceleration and 3-axial angular velocity readings from the gyroscope of a Samsung Galaxy S II at a constant rate of 50Hz accelerometer. The input data consists of 561 which we need to analyze and our goal is to predict if a user is doing any of the following activities: WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING from their smartphone activity.

I will be approaching this as any other multi-class classification problem and after analyzing the dataset and doing all the needful pre-processing on the dataset. I will apply Naive Bayes Algorithm to do an initial classification to check how the algorithm works on the dataset. After that I plan to use a Logistic Regression Classifier, a Support Vector Classifier followed by an ensemble using LightGBM and finally a Keras Neural Network to check which model performs best for the problem at hand. The solution will be classifying a user's activity to any of the following six activities: (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING or LAYING based on the input feature vector from his smartphone's sensor readings.

My approach is to find an optimal model which can solve the task in hand accurately and quickly. So, I will be using few classification algorithms on the dataset to see which one performs better and then use the one that provides the best output and work on that model further to fine tune its parameters to reach an optimum model.

DataSet :

The dataset I am going to use is an open source dataset which can be found in the UCI Machine Learning repository .(http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones).

How data was recorded:

The data set was collected from an experiment which was carried out with a group of 30 volunteers within an age bracket of 19-48 years. They performed a series of activities standing, sitting, lying, walking, walking downstairs and walking upstairs. The experiment also included postural transitions that occurred between the static postures which are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-tolie, and lie-to-stand. All the participants were wearing a smartphone (Samsung Galaxy S II) on the waist during the experiment execution. Data was

captured with 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz using the embedded accelerometer and gyroscope of the device. The experiments were video-recorded to label the data manually. The obtained dataset was randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

Feature names :

These sensor signals are preprocessed by applying noise filters and then sampled in fixed-width windows(sliding windows) of 2.56 seconds each with a 50% overlap. ie., each window has 128 readings.

1. From Each window, a feature vector was obtained by calculating variables from the time and frequency domain.In our dataset, each datapoint represents a window with different readings
2. The acceleration signal was separated into Body and Gravity acceleration signals(**tBodyAcc-XYZ** and **tGravityAcc-XYZ**) using some low pass filter with a corner frequency of 0.3Hz.
3. After that, the body linear acceleration and angular velocity were derived in time to obtain jerk signals (**tBodyAccJerk-XYZ**and **tBodyGyroJerk-XYZ**).
4. The magnitude of these 3-dimensional signals were calculated using the Euclidean norm. This magnitudes are represented as features with names like tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag and tBodyGyroJerkMag.
5. Finally, We've got frequency domain signals from some of the available signals by applying a FFT (Fast Fourier Transform). These signals obtained were labeled with **prefix 'f'** just like original signals with **prefix 't'**. These signals are labeled as **fBodyAcc-XYZ**, **fBodyGyroMag** etc.,.
7. These are the signals that we got so far.

- tBodyAcc-XYZ
- tGravityAcc-XYZ
- tBodyAccJerk-XYZ
- tBodyGyro-XYZ
- tBodyGyroJerk-XYZ
- tBodyAccMag
- tGravityAccMag
- tBodyAccJerkMag
- tBodyGyroMag
- tBodyGyroJerkMag
- fBodyAcc-XYZ
- fBodyAccJerk-XYZ
- fBodyGyro-XYZ
- fBodyAccMag

- fBodyAccJerkMag
- fBodyGyroMag
- fBodyGyroJerkMag

8. We can estimate some set of variables from the above signals. ie., We will estimate the following properties on each and every signal that we recorded so far.

- **mean()**: Mean value
- **std()**: Standard deviation
- **mad()**: Median absolute deviation
- **max()**: Largest value in array
- **min()**: Smallest value in array
- **sma()**: Signal magnitude area
- **energy()**: Energy measure. Sum of squares divided by the number of values.
- **iqr()**: Interquartile range
- **entropy()**: Signal entropy
- **arCoeff()**: Autoregression coefficients with Burg order equal to 4
- **correlation()**: correlation coefficient between two signals
- **maxInds()**: index of the frequency component with largest magnitude
- **meanFreq()**: Weighted average of the frequency components to obtain a mean frequency
- **skewness()**: skewness of the frequency domain signal

9. We can obtain some other vectors by taking the average of signals in a single window sample. These are used on the angle() variable'

- **gravityMean**
- **tBodyAccMean**
- **tBodyAccJerkMean**
- **tBodyGyroMean**
- **tBodyGyroJerkMean**
- **kurtosis()**: kurtosis of the frequency domain signal
- **bandsEnergy()**: Energy of a frequency interval within the 64 bins of FFT of each window.
- **angle()**: Angle between two vectors.

Data Exploration :

The data-set consists of the following features:

• A 561-feature vector with time and frequency domain variables.

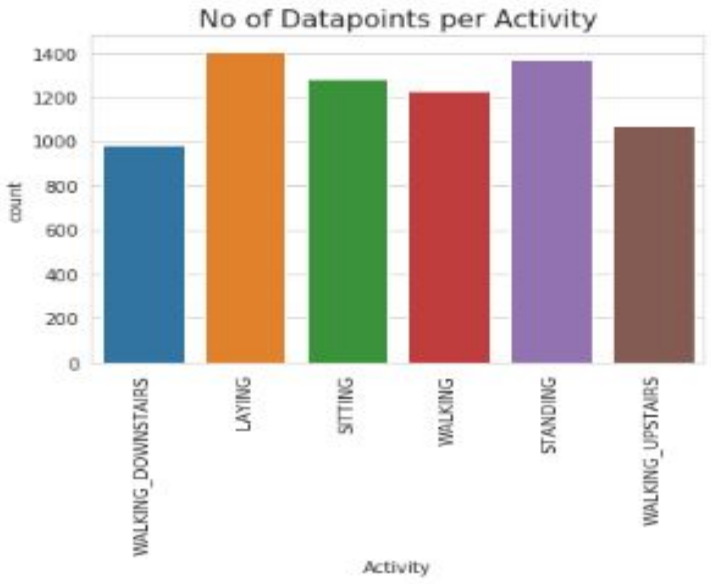
• Its associated activity labels (WALKING, WALKING_UPSTAIRS, WALKING_ DOWNSTAIRS, SITTING, STANDING or LAYING).

• An identifier of the subject who carried out the experiment.

Data Cleaning:

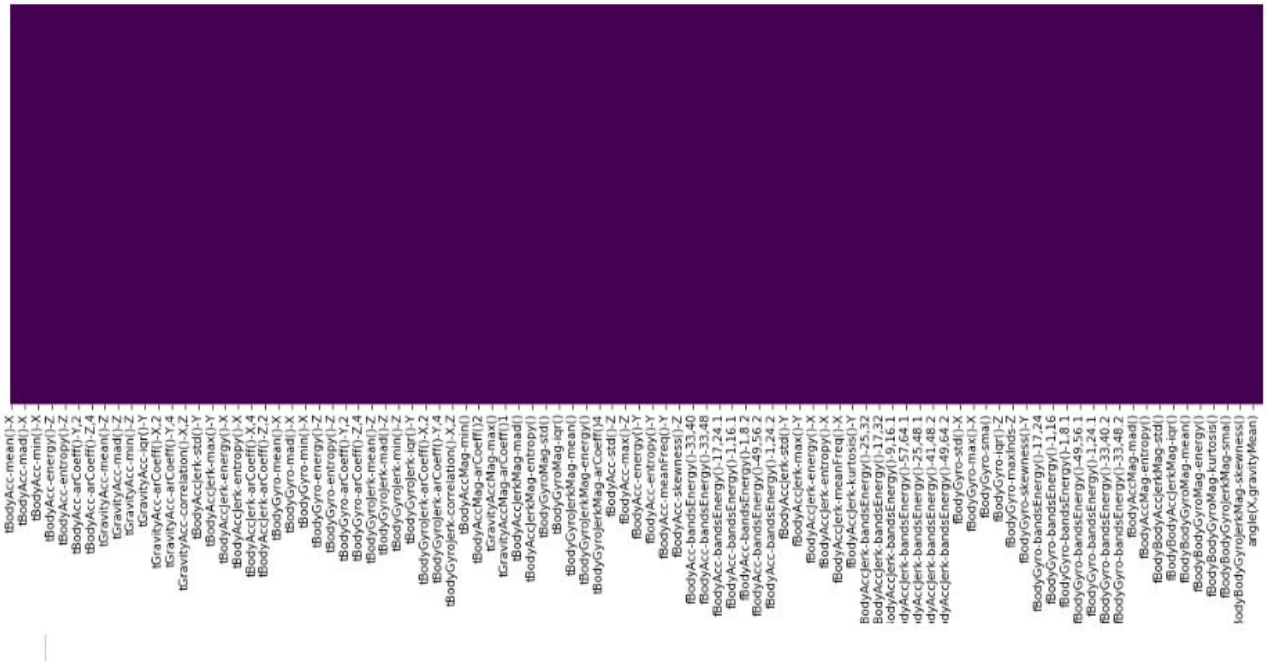
INITIAL EXPLORATION AND VISUALIZATION: In-order to make sure the dataset under consideration doesn't have any major data quality issues like class imbalance, missing data etc. that will hinder the performance of classification models to a huge extent we start with the basic analysis of our dataset using visualizations.

After importing the necessary libraries like NumPy, pandas, matplotlib and seaborn we start our analysis of data by checking if the data has class imbalance issue. The plot of the target variable 'Activity'. We observed that there isn't much variation in the class label counts.



Missing Data:

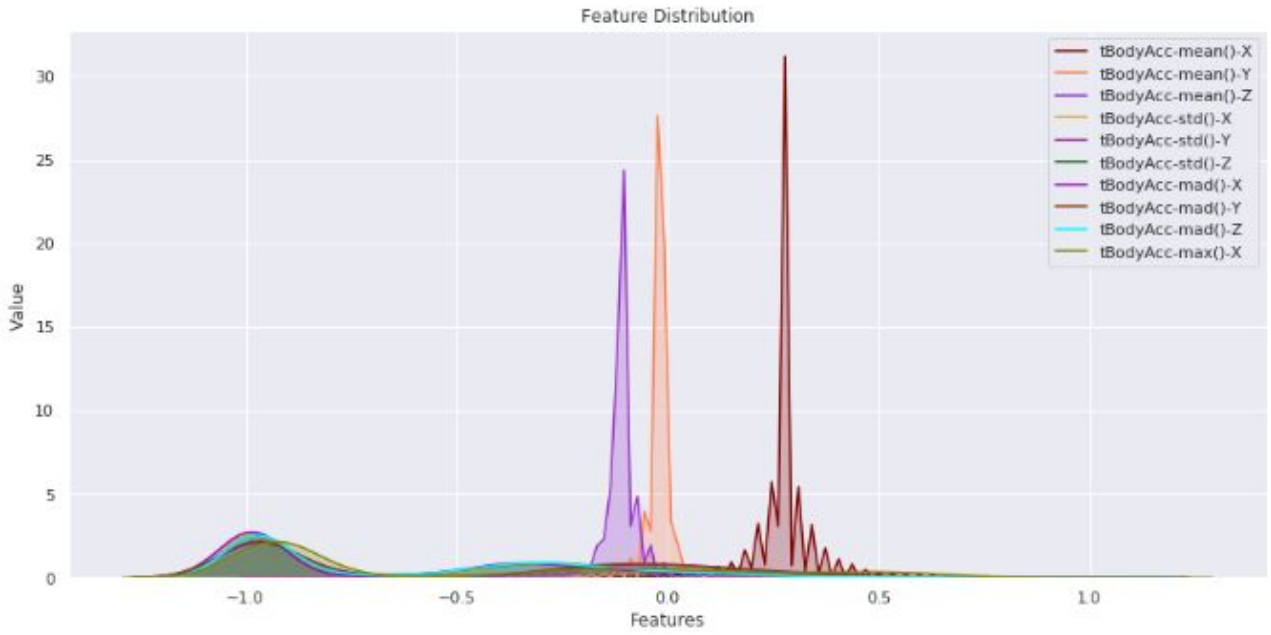
To check for missing data points, we have generated a heatmap which would show us missing instances as colored dash lines in each feature. We observed that there are no missing

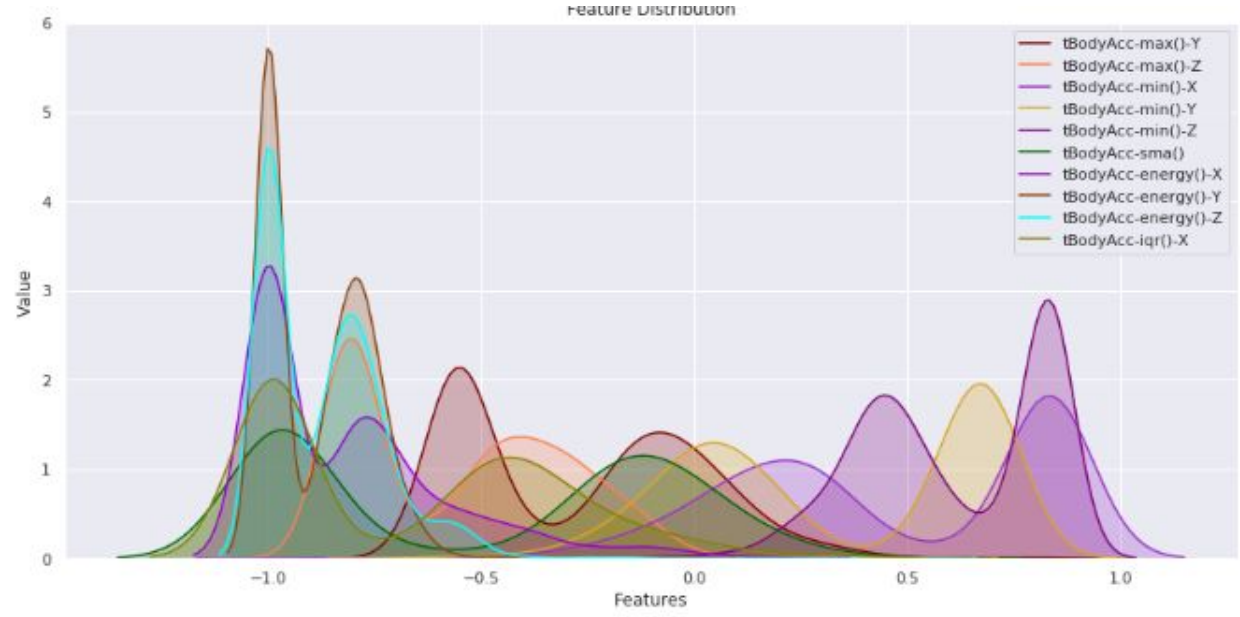


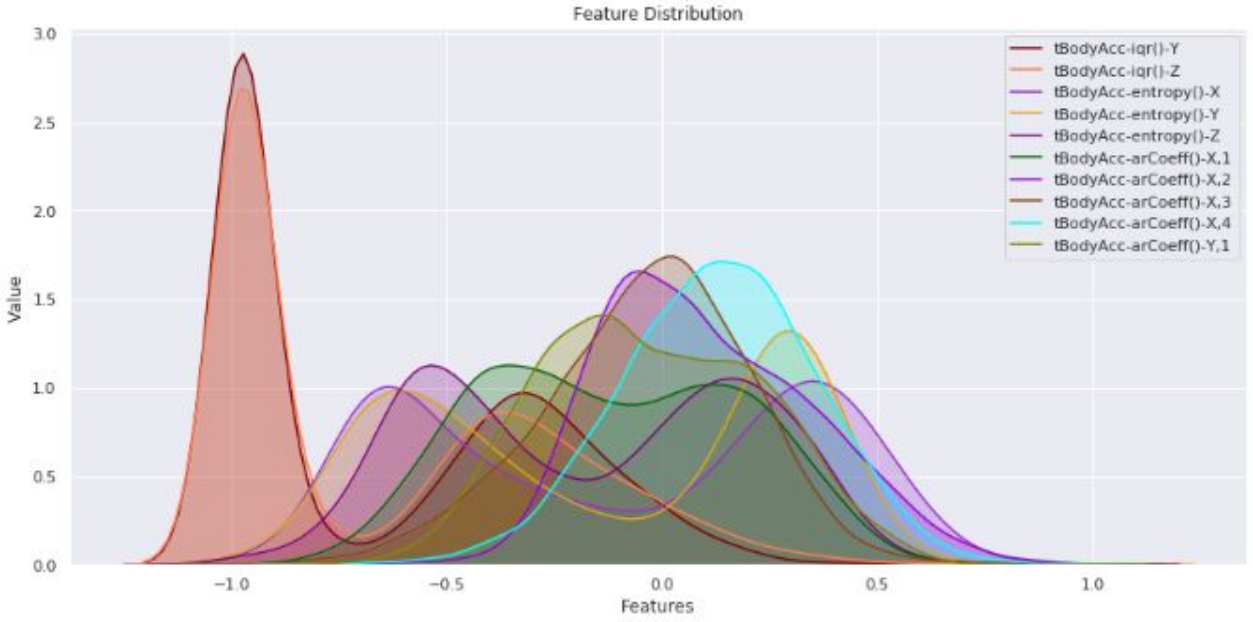
Static and Dynamic Activities :

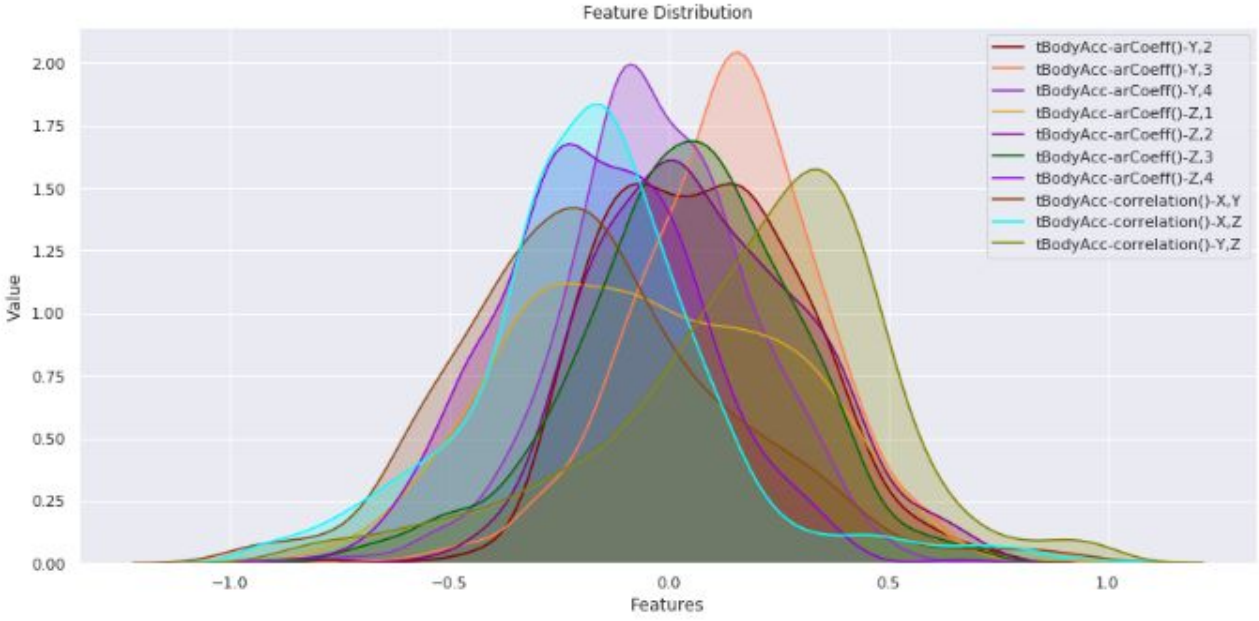
In static activities (sit, stand, lie down) motion information will not be very useful.In the dynamic activities (Walking, WalkingUpstairs,WalkingDownstairs) motion info will be significant.

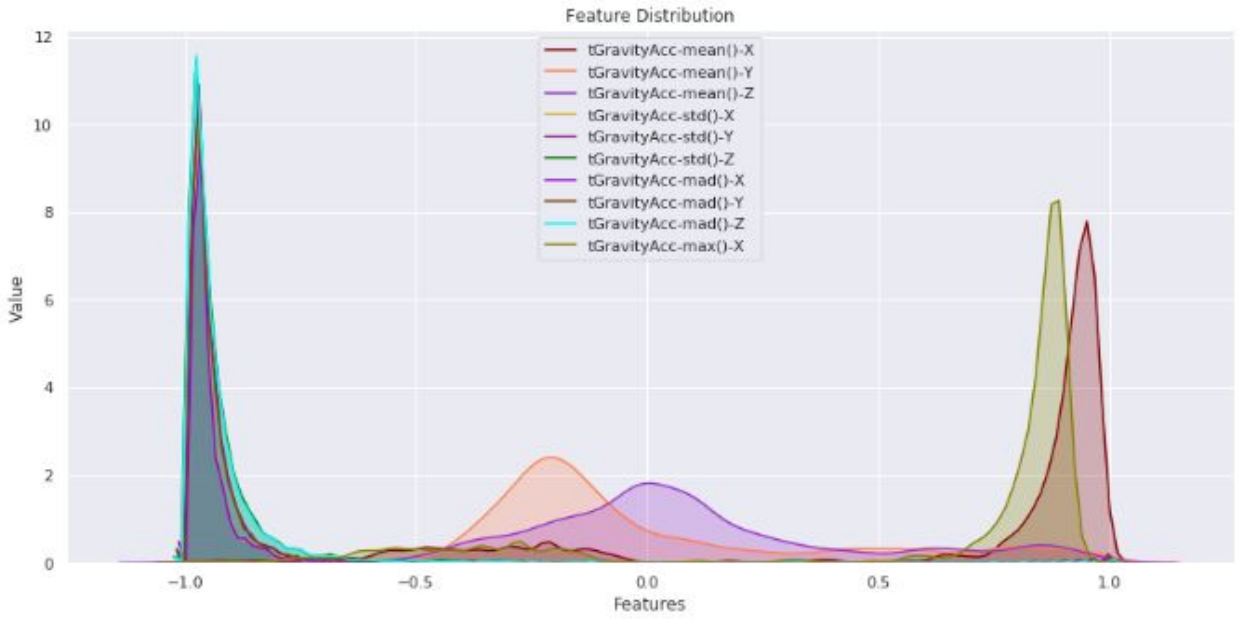
Feature visualization :


Feature Distribution
tBodyAcc-mean()-X
tBodyAcc-mean()-Y
tBodyAcc-mean()-Z
tBodyAcc-std()-X
tBodyAcc-std()-Y
tBodyAcc-std()-Z
tBodyAcc-mad()-X
tBodyAcc-mad()-Y
tBodyAcc-mad()-Z
tBodyAcc-max()-X
Value
Features
30
25
20
15
10
5
0
−1.0
−0.5
0.0
0.5
1.0


tBodyAcc-max()-Y
tBodyAcc-max()-Z
tBodyAcc-min()-X
tBodyAcc-min()-Y
tBodyAcc-min()-Z
tBodyAcc-sma()
tBodyAcc-energy()-X
tBodyAcc-energy()-Y
tBodyAcc-energy()-Z
tBodyAcc-iqr()-X
Value
Features
6
5
4
3
2
1
0
−1.0
−0.5
0.0
0.5
1.0

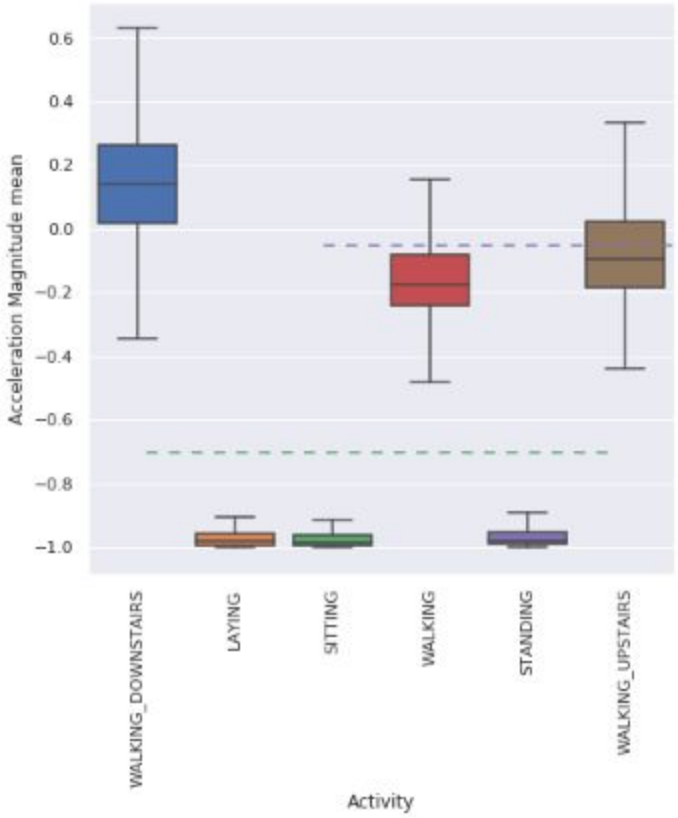

Feature Distribution
tBodyAcc-iqr()-Y
tBodyAcc-iqr()-Z
tBodyAcc-entropy()-X
tBodyAcc-entropy()-Y
tBodyAcc-entropy()-Z
tBodyAcc-arCoeff()-X,1
tBodyAcc-arCoeff()-X,2
tBodyAcc-arCoeff()-X,3
tBodyAcc-arCoeff()-X,4
tBodyAcc-arCoeff()-Y,1
Value
Features
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
1.0


Feature Distribution
tBodyAcc-arCoeff()-Y,2
tBodyAcc-arCoeff()-Y,3
tBodyAcc-arCoeff()-Y,4
tBodyAcc-arCoeff()-Z,1
tBodyAcc-arCoeff()-Z,2
tBodyAcc-arCoeff()-Z,3
tBodyAcc-arCoeff()-Z,4
tBodyAcc-correlation()-X,Y
tBodyAcc-correlation()-X,Z
tBodyAcc-correlation()-Y,Z
Value
Features
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1.0
−0.5
0.0
0.5
1.0



Relationship between Magnitude of an acceleration and Activity

If tBodyAccMag-mean is < -0.8 then the Activities are either Standing or Sitting or Laying. If tBodyAccMag-mean is > -0.6 then the Activities are either Walking or WalkingDownstairs or WalkingUpstairs. If tBodyAccMag-mean > 0.0 then the Activity is WalkingDownstairs. We can classify 75% the Acitivity labels with some errors.



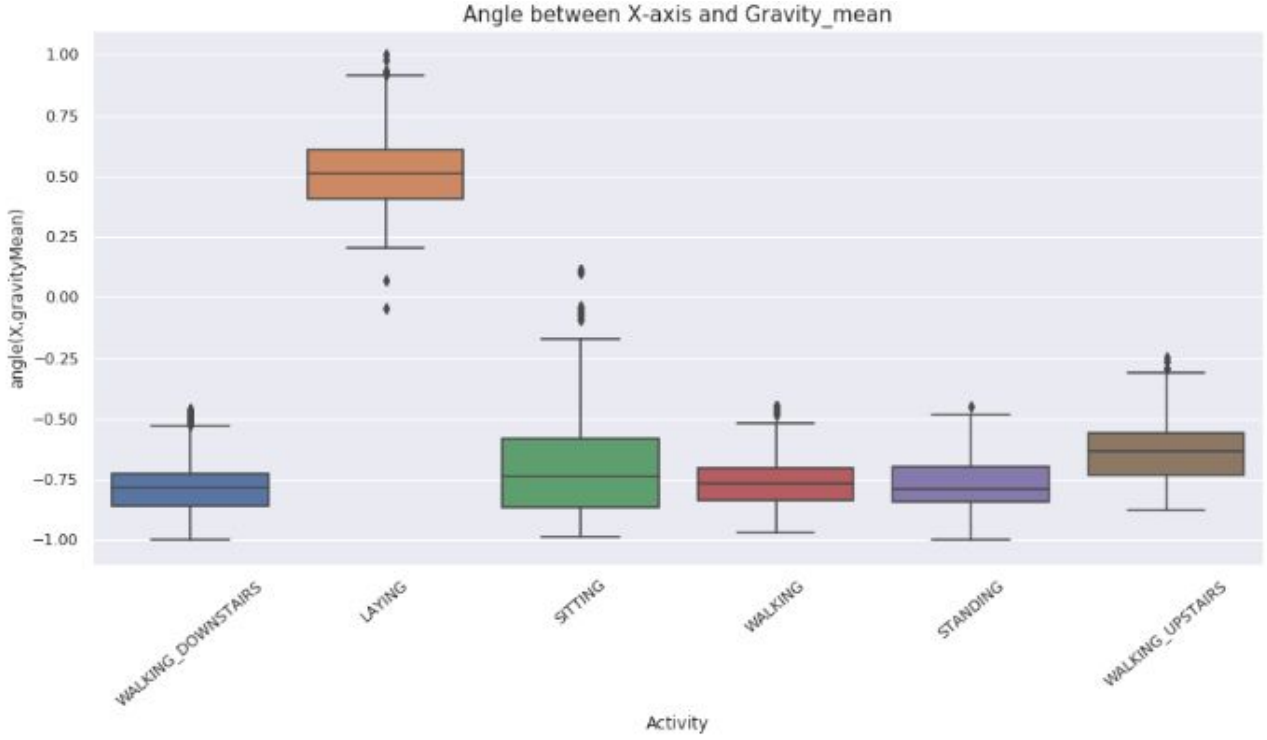
If tBodyAccMag-mean is < -0.8 then the Activities are either Standing or Sitting or Laying.
If tBodyAccMag-mean is > -0.6 then the Activities are either Walking or WalkingDownstairs or WalkingUpstairs.

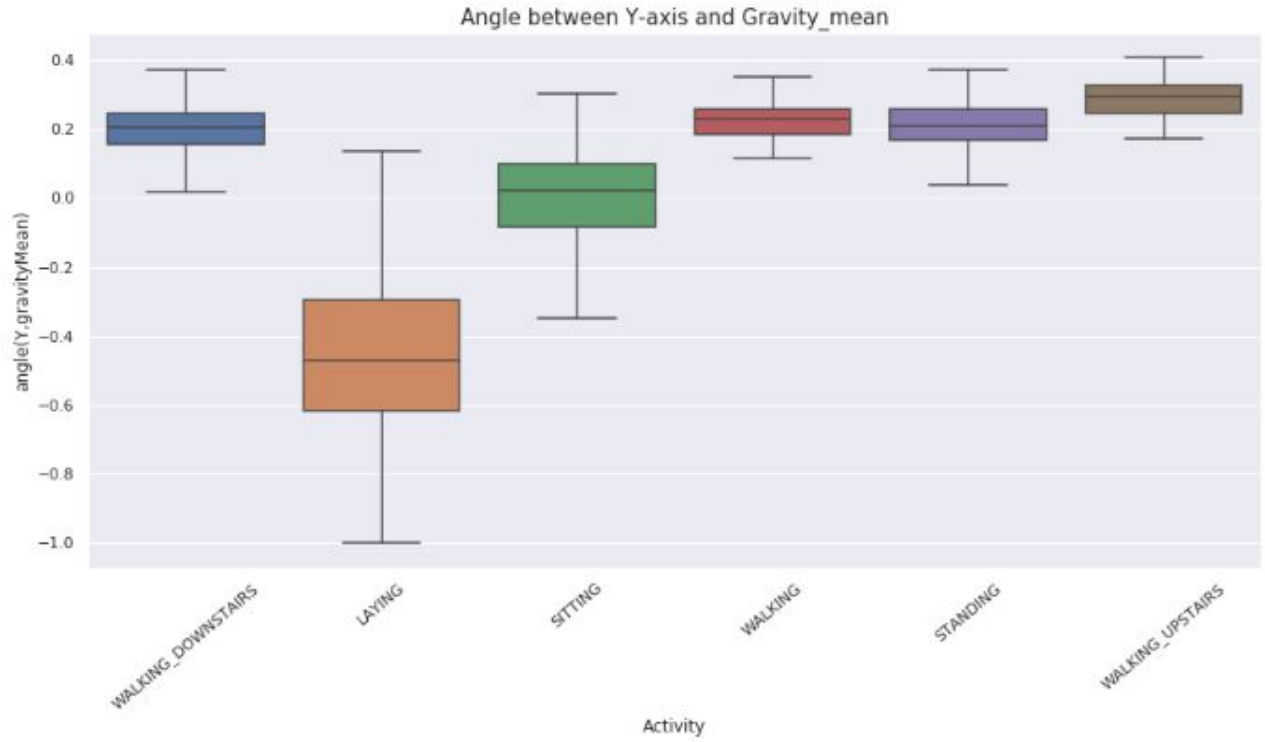
If tBodyAccMag-mean > 0.0 then the Activity is WalkingDownstairs.

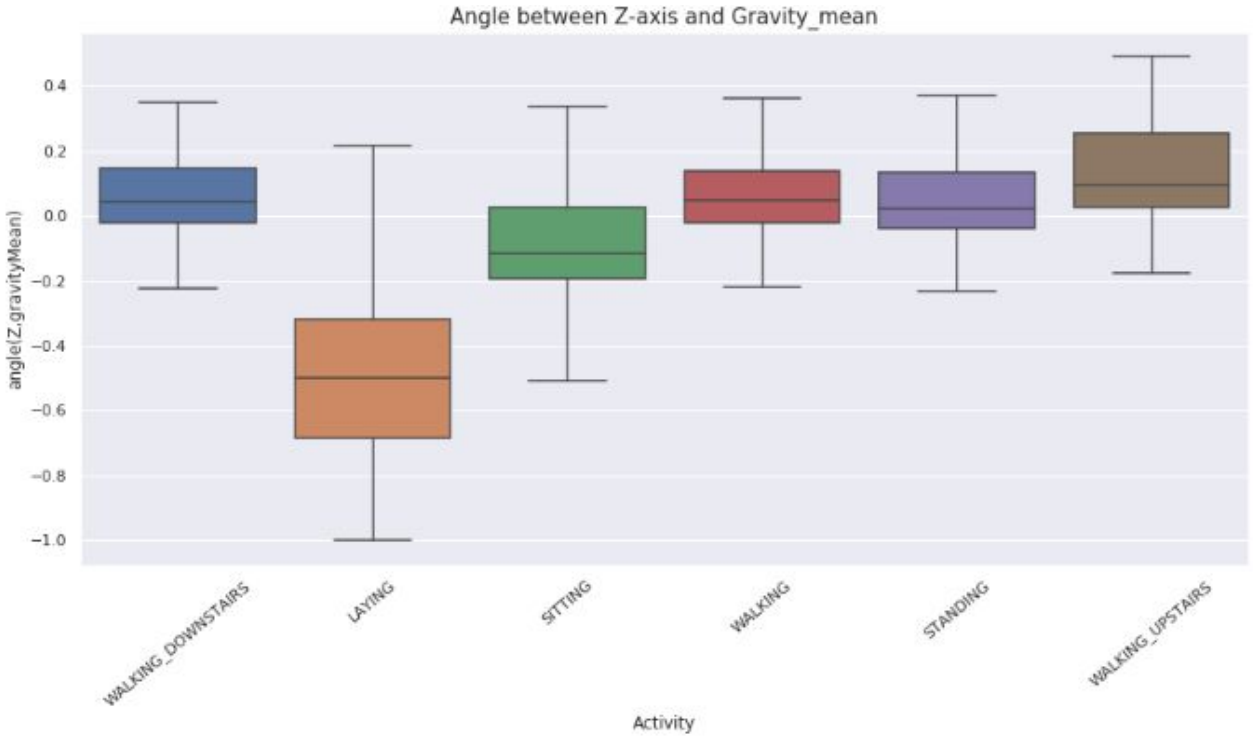
We can classify 75% the Acitivity labels with some errors.

Relationship between Position of Gravity Acceleration Components and the Target (activity) :

If angleX,gravityMean > 0 then Activity is Laying.We can classify all data points belonging to Laying activity with just a single if else statement.

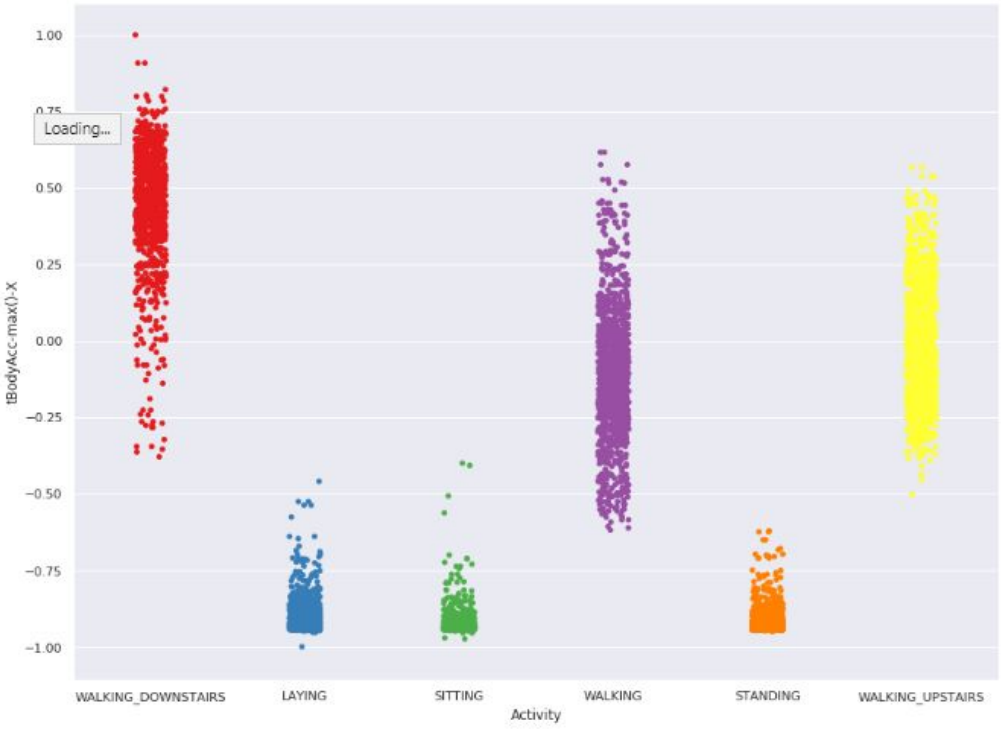

Angle between X-axis and Gravity_mean
angle(X,gravityMean)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
WALKING_DOWNSTAIRS
LAYING
SITTING
WALKING
STANDING
WALKING_UPSTAIRS
Activity

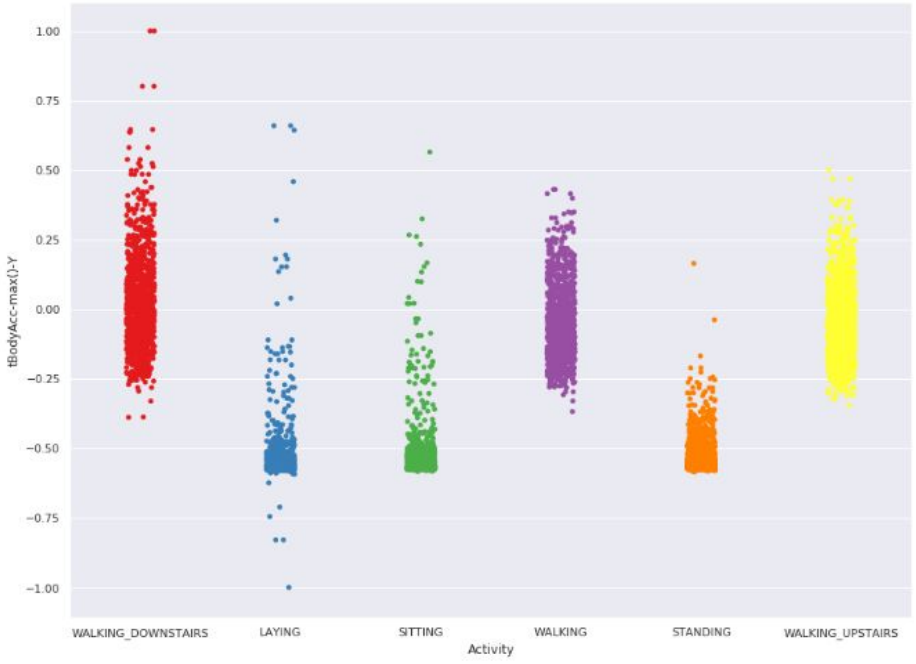

Angle between Y-axis and Gravity_mean
angle(Y,gravityMean)
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
WALKING_DOWNSTAIRS
LAYING
SITTING
WALKING
STANDING
WALKING_UPSTAIRS
Activity

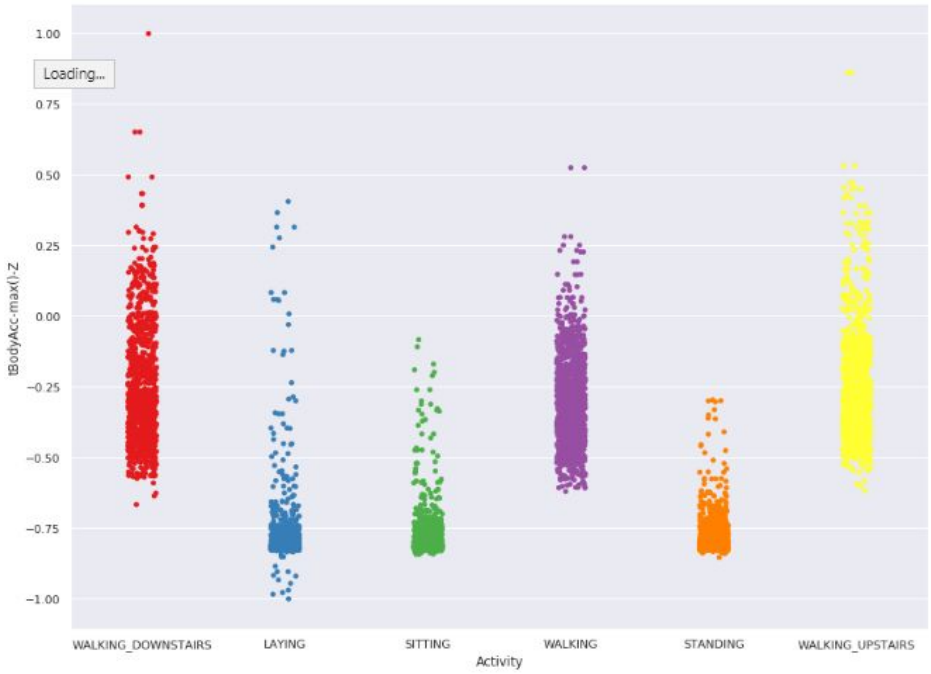


From the above visualization we can see that the Gravity Acceleration Component in X direction is more variable than in Y and Z direction. If we observe closely, for the active activities: WALKING, WALKING_UPSTAIRS and WALKING_DOWNSTAIRS the variation in X-axis more compared the passive activities: STANDING, SITTING and LAYING.

Relationship between Max body acceleration and the Target (activity) :





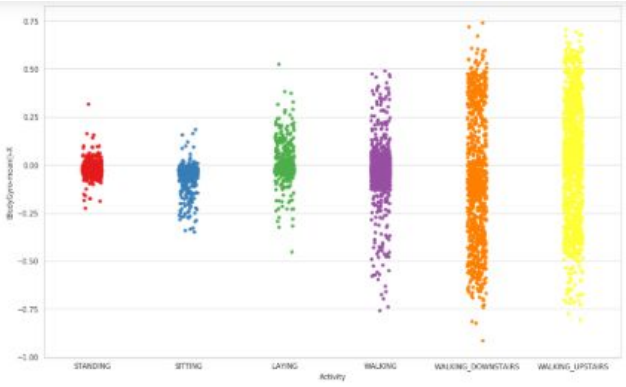


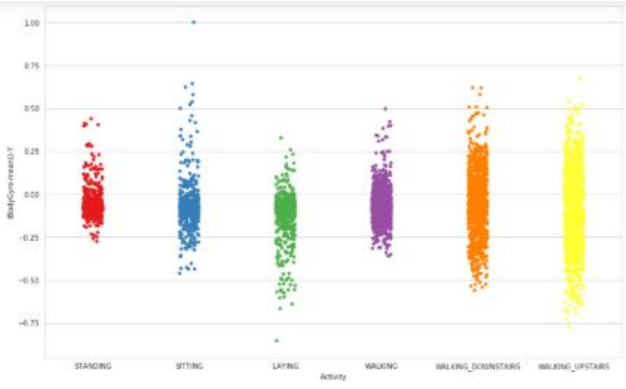
Comparing the above Strip-plot, we can clearly see that for Max Acc in all directions, the Passive activities are way below the active ones and for the active one the variation is also huge. If we notice in the first of the two plots, WALKING_DOWNSTAIRS has a greater value of Max body acceleration. This can be explained as when a person walks down the stairs, his body moves comparatively faster compared to when he is steadily walking or Walking_upstairs.

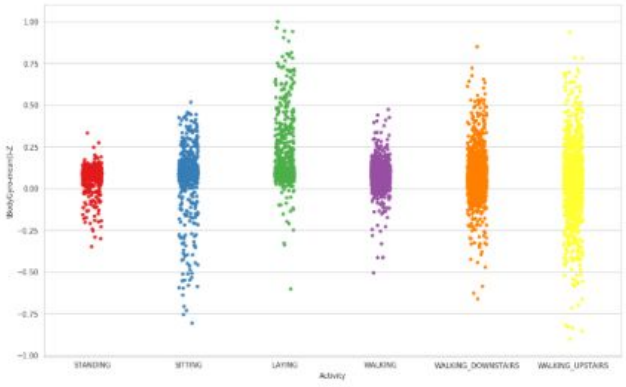
Analysis indicates that there is a clear distinction in the maximum values between the passive and active activities as all the passive activities fall below the active ones. The analysis from above plots reveals that values along the X-axis can help us differentiate between walking, walking upstairs and walking downstairs but does not provide any insights into the passive activities. This provides a certain indication that the acceleration alone is not sufficient enough for ambulatory activity recognition but data from a different sensor such as a gyroscope would help in differentiating among the passive activities.

Relationship between Gyroscope body mean and the Target (activity) :

The plot does not show any clear distinction between the activities. In a similar way, other features engineered may provide important insights into recognizing human activities.


STANDING
SITTING
LAYING
WALKING
WALKING_DOWNSTAIRS
WALKING_UPSTAIRS
Activity


STANDING
SITTING
LAYING
WALKING
WALKING_DOWNSTAIRS
WALKING_UPSTAIRS
Activity


STANDING
SITTING
LAYING
WALKING
WALKING_DOWNSTAIRS
WALKING_UPSTAIRS
Activity

Algorithms and Techniques:

The problem in question is a multi-class Classification problem. The input data consists 3-axial linear acceleration and 3-axial angular velocity readings from the gyroscope of a Samsung Galaxy S II at a constant rate of 50Hz accelerometer. The input data consists of 561 which we need to analyze and our goal is to predict if a user is doing any of the following activities: WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING from their smartphone activity. I will be approaching this as any other multi-class classification problem

For the pre-processing step, as our dataset is already normalized and scaled in the range [1, 1], we can move on to feature selection step. We can see that the 'subject' column which is the subject who has taken reading for this dataset has no real effect on the output activity as all the subjects has performed the same activities and predicting an activity is not dependent on the subject in any way, we can discard that feature. Before starting to work on the model, I will first do some visualization on the dataset to see which features are more important and strongly related. After removing 'Subject' and storing 'Activity' separately, we will be left with 561 features, and I plan to use all of these features while training and predicting with our model.
The main goal of this project is to find the model which best performs with our problem and for that the Machine Learning algorithms we consider are:

1) Logistic Regression
2) Support Vector Machines (SVM)
3) k – Nearest Neighbor (kNN)
4) Random Forests
5) Decision Trees
6) Artificial Neural Network

I will compare the accuracy and performance of these models using confusion matrix.

Metrics :

Once the model has been trained on the training data, its performance will be evaluated using the test data.

The following metrics will be used:

Accuracy – will be used for evaluating the performance of the model on the test data.

Confusion Matrix - will be used in order to compare the model with the Benchmark model. A *confusion matrix* is used to describe the performance of a classification model.

TOTAL	**PREDICTED: YES**	**PREDICTED: NO**
ACTUAL: YES	True Positives	False Negatives
ACTUAL: NO	False Positives	True Negatives

True Positives (TP) – The model predicted 'YES' and the prediction was *True*

False Positives (FP) – The model predicted 'YES' but the prediction was *False*

False Negatives (FN) – The model predicted 'NO' but the prediction was *False*

True Negatives (TN) – The model predicted 'NO' and the prediction was *True* +

Accuracy is the most common evaluation metric used for classification problems. In particular, accuracy is very useful when there are an **equal number of samples in each class**. Since our dataset have similar characteristics, accuracy would be a suitable metric to evaluate the model.

Logistic regression is basically a regression method with the exception that it provides probability output of each class rather than actual predicted class. Classification using logistic regression is a supervised learning method, and therefore requires a labeled dataset. Our problem is a multiclass classification problem

```
Classification Report
                    precision    recall  f1-score   support

           LAYING       1.00      1.00      1.00       537
          SITTING       0.97      0.88      0.92       491
         STANDING       0.90      0.97      0.94       532
          WALKING       0.94      1.00      0.97       496
WALKING_DOWNSTAIRS       1.00      0.97      0.98       420
  WALKING_UPSTAIRS       0.97      0.95      0.96       471

         accuracy                           0.96      2947
        macro avg       0.96      0.96      0.96      2947
     weighted avg       0.96      0.96      0.96      2947

Confusion Matrix
[[537   0   0   0   0   0]
 [  0 432  55   0   0   4]
 [  0  13 517   2   0   0]
 [  0   0   0 494   2   0]
 [  0   0   0   4 407   9]
 [  0   0   0  23   0 448]]
Accuracy Score = 0.9619952494061758
```

Random Forest Classifier:

Random Forest Classifier builds a forest which is an ensemble of decision trees. It creates a set of decision trees from a randomly selected subset of the training data and aggregates the decision from all the trees to decide the final output. This technique is robust as it prevents the noisy output of some trees affecting the final decision and avoids overfitting. It cancels out the bias by averaging all the predictions. Random forest is differentiated from decision trees as it does not search for the best feature while splitting the node. It instead searches for the most appropriate feature from a subset of features. This provides diversity and randomness to the algorithm. The algorithm can be easily modeled to both classification and regression problems.

```
Random Forest Accuracy: 0.92026
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       537
           1       0.92      0.88      0.90       491
           2       0.89      0.92      0.91       532
           3       0.87      0.97      0.92       496
           4       0.96      0.83      0.89       420
           5       0.89      0.89      0.89       471

    accuracy                           0.92      2947
   macro avg       0.92      0.92      0.92      2947
weighted avg       0.92      0.92      0.92      2947

Confusion Matrix
[[537   0   0   0   0   0]
 [  0 432  59   0   0   0]
 [  0  40 492   0   0   0]
 [  0   0   0 481  10   5]
 [  0   0   0  25 349  46]
 [  0   0   0  44   6 421]]
Accuracy Score = 0.9202578893790295
```

Decision Tree Classifier:

A popular machine learning model, decision trees uses a tree-like structure to represent decisions. They are constructed in a top-down structure with the use of metrics such as Gini impurity and information. It calculates the importance of each feature and uses it to split the elements into homogenous subsets. The nodes represent the condition of the split and the leaf nodes represent the decision or the predicted output. The branches or the edges of the tree directly to one of the output variables. Decision trees are modeled for both classification and regression problems. Though the decision tree is easy to understand it tends to overfit as it continues to split on attributes and trains critically on the training data. To avoid overfitting the decision tree is generally pruned to stop it from growing too deep.

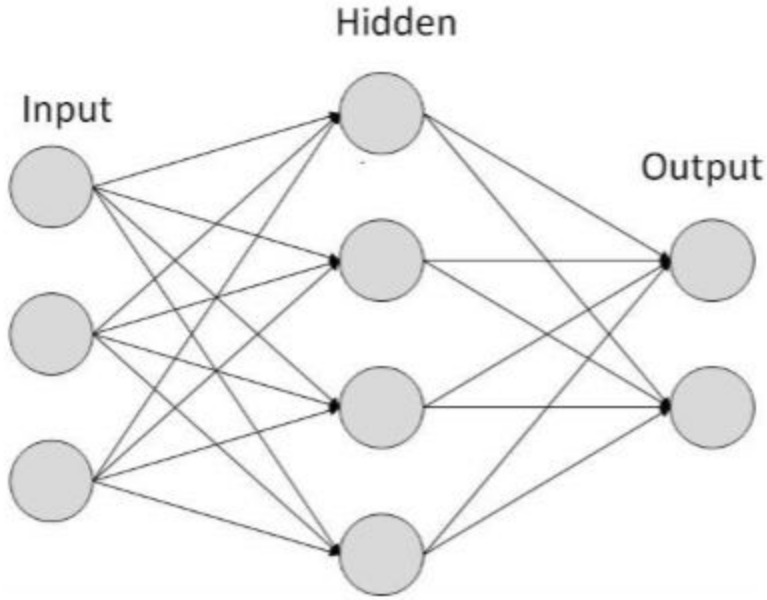
```
Classification Report
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       183
           1       0.82      0.68      0.74       170
           2       0.75      0.85      0.80       178
           3       0.70      0.65      0.67       185
           4       0.74      0.90      0.81       134
           5       0.56      0.50      0.53       149

    accuracy                           0.77       999
   macro avg       0.76      0.76      0.76       999
weighted avg       0.76      0.77      0.76       999

Confusion Matrix
[[183   0   0   0   0   0]
 [  3 116  51   0   0   0]
 [  0  26 152   0   0   0]
 [  0   0   0 120   9  56]
 [  0   0   0  11 120   3]
 [  0   0   0  41  33  75]]
Accuracy Score = 0.7667667667667668
```

Support Vector Machine

The next supervised machine learning model which I have applied to my dataset is Support Vector Machine. This algorithm outputs an optimal hyperplane dividing plane into two parts wherein each class can lay on the other side. In my dataset, Support Vector Machine divides hyperplane into 6 parts and provide optimal hyperplane. The accuracy achieved by using Support Vector Machine is 94%.

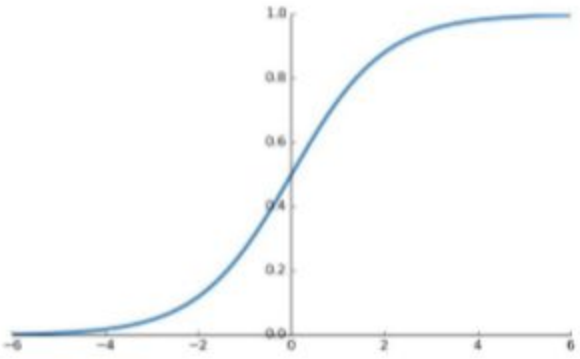
Artificial Neural Network:

Artificial Neural Network is a system inspired by the biological neuron structure of the brain. The structure can be defined as a set of connected neurons organized in consecutive layers. The input layer acts as the first layer which brings in the data into the network. The hidden layer consists of artificial neurons which take in a set of weighted inputs and apply an activation function to produce an output. There could be multiple hidden layers in a network which makes it capable to solve complex problems. The output of a layer of neurons is passed on as the input to the successive layer and is termed as a feed forward network. The output layer provides the final predictive output.



Activation functions core logic of the neural networks and defines the output of the neuron given an input or a set of inputs. Following are the different activation functions:

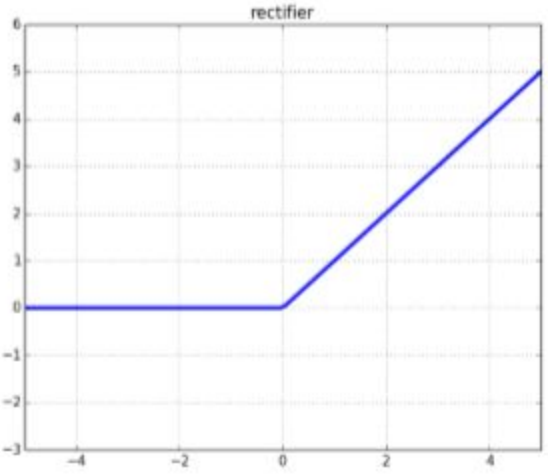
• **Sigmoid**: The sigmoid function has a characteristic 'S' shaped curved and is widely used in binary classification. The function generates a probability output between 0 and 1 for a set of input.



● **ReLU:**

Rectified Linear Units are most commonly used in the hidden layers of the artificial neural networks. The function is such that if the input is less than zero the output is 0 and if the input is greater than zero it gives the input itself as the output.

$$ReLU(x) = \max(0, x)$$



● **Softmax:**

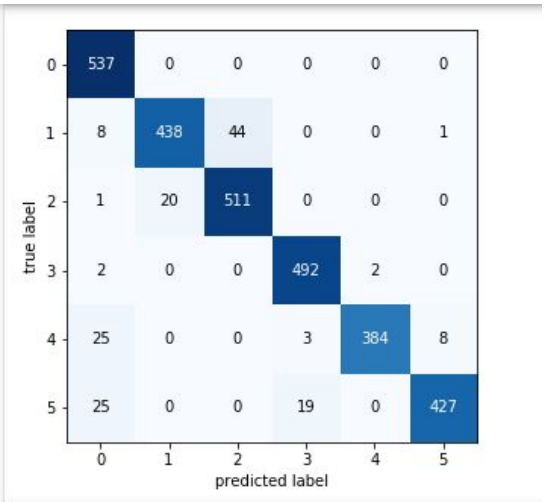
The softmax activation function is used for multi-class classification. It calculates the probability distribution of each class over all possible target classes. Based on the calculated probabilities it determines the output for a given set of inputs.

$$Softmax(x) = \frac{e^j}{\sum_i e^i}$$

Supervised Neural Network using multi-layer preceptron

We have used nnet package MLP CLassifier to train our dataset, which is used for feed-forward neural networks with a single hidden layer of 90 nodes. The nnet package trains the artificial neural network using backpropagation method. In this method error at the output is determined and then it is propagated back into the network. To minimize the error resulting from each neuron, the weights are updated . A simple MLP classifier using Stochastic gradient descent optimizer gave an accuracy of 95%.

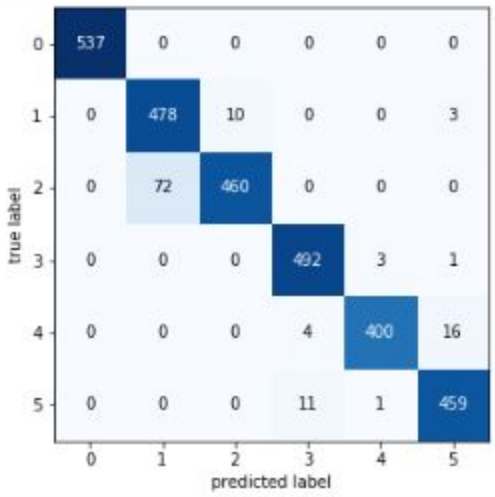
	precision	recall	f1-score	support
0	0.90	1.00	0.95	537
1	0.96	0.89	0.92	491
2	0.92	0.96	0.94	532
3	0.96	0.99	0.97	496
4	0.99	0.91	0.95	420
5	0.98	0.91	0.94	471
accuracy			0.95	2947
macro avg	0.95	0.94	0.95	2947
weighted avg	0.95	0.95	0.95	2947



Neural Network Model Using Keras

The artificial neural network was built using the Keras library using Tensorflow as its backend. The network was configured to have two hidden layers with 64 hidden units in each with the "ReLU" activation function for each neuron. The output layer was configured to utilize a 'Softmax' activation and the 'Adam' and SGD optimizer were used to boost accuracy. The model was compiled to run over 500 epochs with a batch size of 128 using 'Categorical Cross Entropy' as its loss function.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	537
1	0.87	0.97	0.92	491
2	0.98	0.86	0.92	532
3	0.97	0.99	0.98	496
4	0.99	0.95	0.97	420
5	0.96	0.97	0.97	471
accuracy			0.96	2947
macro avg	0.96	0.96	0.96	2947
weighted avg	0.96	0.96	0.96	2947



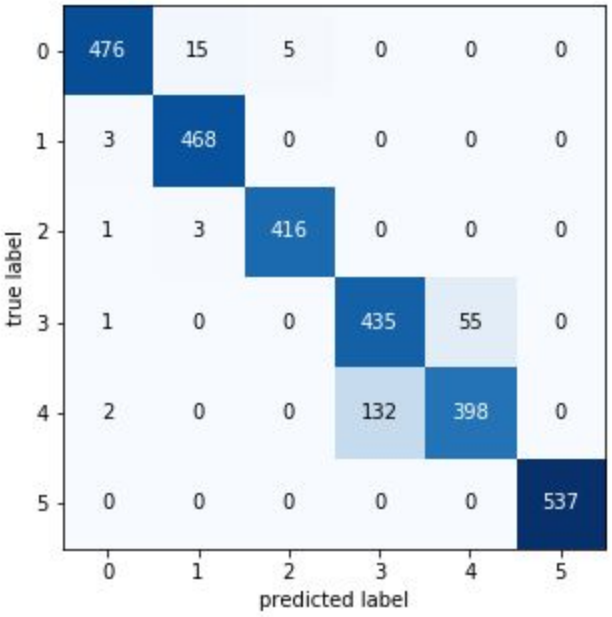
LSTM Network Model

LSTM network models are a type of recurrent neural network that are able to learn and remember over long sequences of input data. They are intended for use with data that is comprised of long sequences of data, up to 200 to 400 time steps. They may be a good fit for this problem.

The model can support multiple parallel sequences of input data, such as each axis of the accelerometer and gyroscope data. The model learns to extract features from sequences of observations and how to map the internal features to different activity types.

The benefit of using LSTMs for sequence classification is that they can learn from the raw time series data directly, and in turn do not require domain expertise to manually engineer input features. The model can learn an internal representation of the time series data and ideally achieve comparable performance to models fit on a version of the dataset with engineered features.

The RNN-LSTM model is built using the Keras library and is defined as a Sequential Keras model. The model first has a single hidden LSTM layer which is used to extract features from a sequence of input data. A dropout layer is added to the model to reduce overfitting on the training data. Next, a fully connected dense layer is added to the model which interprets the features followed by an output layer which gives out the final predictions of the human activity. 'Categorical Cross Entropy' is used as a loss function along with an 'Adam' optimizer to boost accuracy and optimize the network. 'Categorical Cross Entropy' is a commonly used loss function for classification tasks while the 'Adam' optimizer is well suited for large datasets as they are computationally efficient and require less memory. The 'ReLU' activation function is used in the dense layer with a 'Softmax' activation function for the output. The ReLU activation function overcomes the trouble of vanishing gradients and is hence preferred over Sigmoid and Tanh activation functions. The 'Softmax' activation function provides a probability distribution over all the classes and reacts well to low and high simulation. The model is run for a total of 15 epochs with a batch size of 64 samples.
We can see that the model performed well, achieving a classification accuracy of about 93% trained on the raw dataset



	precision	recall	f1-score	support
0	0.99	0.96	0.97	496
1	0.96	0.99	0.98	471
2	0.99	0.99	0.99	420
3	0.77	0.89	0.82	491
4	0.88	0.75	0.81	532
5	1.00	1.00	1.00	537
accuracy			0.93	2947
macro avg	0.93	0.93	0.93	2947
weighted avg	0.93	0.93	0.93	2947