## INTRODUCTION:

**Human Activity Recognition** is one of the major advancements in history of computer science since its inception in the utilizing body sensors to detect activities. Latest Smart-Phones are embedded with a wide range of sensors such as accelerometers, gyroscopes and other inertial sensors which help in monitoring the daily activities of a user. The dataset in consideration was extracted by making use of Sensor based Multi-User Activity Recognition, which in short is recognizing activities of different users using on-body sensors.

Due to the inherent noisy nature of the input, pre-processing of the sensor signals by noise filters was done and **128 reading** were taken within a window of **2.56 seconds**. For each of the window mentioned above, a vector of features along with feature mapping variables like **mean, correlation, signal magnitude area and auto regressor** coefficients were employed. New set of features such as energy of different frequency bands, frequency skewness and angle between vectors were also employed to improve the learning performance. After the above pre-processing of the data into normalized dataset, a 10299 x 562 dataset (561 regressors) was obtained which will further be split into test and training datasets through which we will test the accuracy of different models in predicting the activity of user based on the 561 features and suggest an efficient model with optimal parameter tuning along with further work that could improve the analysis.

## DATA DESCRIPTION:

The dataset has been created using inertial data from the smartphone accelerometers and gyroscopes, targeting the recognition of six different activities, classified as:
Standing-**1**, Sitting-**2**, Laying Down-**3**, Walking-**4**, Walking Downstairs-**5**, Walking Upstairs-**6**

- No. of Observations(n): **10299**
- No. of Features(p): **561**

**Type of Problem: Classification**

For each record in the dataset the following is provided:

- **Triaxial acceleration** from the accelerometer (total acceleration)
- Estimated body acceleration.
- **Triaxial Angular velocity** from the gyroscope.

These three measurements have been collected in a timed window on each participant multiple times. So, we can say that at an instance of time there are three measurements and then so on for all the time instances. That is how we got such a big matrix of data.
Activity label (Standing, Sitting, Laying Down, Walking, Walking Downstairs, Walking Upstairs).

## METHODOLOGY:

### 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 'class' (Fig 1.1), we observed that there isn't much variation in the class label counts.

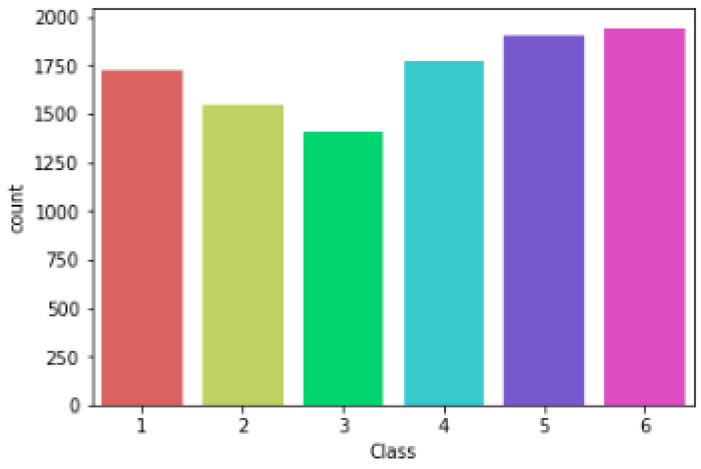


Fig 1.1

**Missing Data:**
To check for **missing data** points, we have generated a heatmap (Fig 1.2) which would show us missing instances as colored dash lines in each feature. We observed that there are no missing values in our dataset and we are all good to go as far as missing values are concerned.

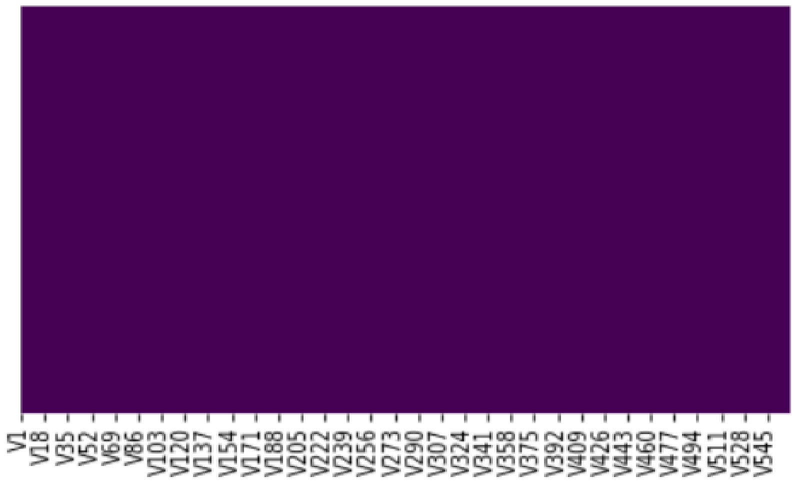


Fig 1.2

Apart from the above analysis we even need to check if the dataset chosen suits our choice of classifiers. Since we will consider models like KNN, SVM etc. to train our data, **Data Normalization** is highly recommended since these classifiers are heavily dependent on distance between different data points and their units of measurement. So, a **3d plot** has been constructed (Fig 1.3) from which we can clearly observe all the data is within the limits of **-1 to 1**. The cluster visualization even confirms the good sparsity of dataset which helps us tackle the problem of dimensionality.

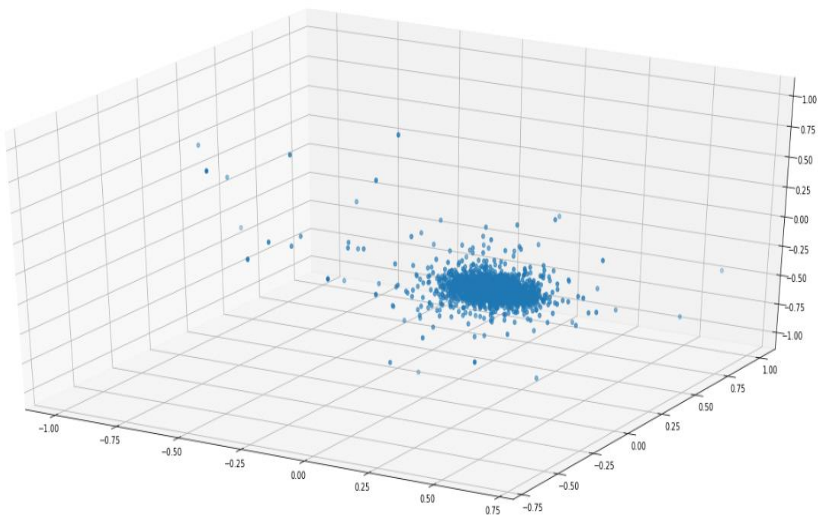
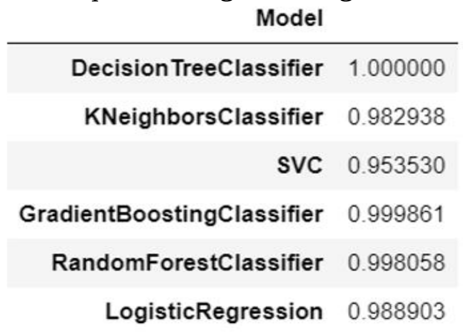


Fig 1.3

**MODEL FITTING:**

Our main goal was to find a model that predicts the target variable with higher efficiency. So, we started our analysis by running all the basic machine learning classifiers i.e. Decision Tree, SVM, RFC, KNN, Logistic Regression, Gradient Boost to check the accuracy scores and interpret results. Based on this approach we started analyzing whether our data is linearly separable or need non-linearly. The following bar graph represents the training errors of all the classifiers from which we will get the idea that what classifier is performing well on the training data.

From Fig 2.1 & Fig 2.2 all the classifiers almost fit the data and there is no **underfitting** visible. This might be due to the reason that classifiers such as gradient boost, decision trees and linear SVM are quite strong in fitting this data as this data is not linearly complex to fit.



| Model | |
|---|---|
| DecisionTreeClassifier | 1.000000 |
| KNeighborsClassifier | 0.982938 |
| SVC | 0.953530 |
| GradientBoostingClassifier | 0.999861 |
| RandomForestClassifier | 0.998058 |
| LogisticRegression | 0.988903 |

Fig 2.1

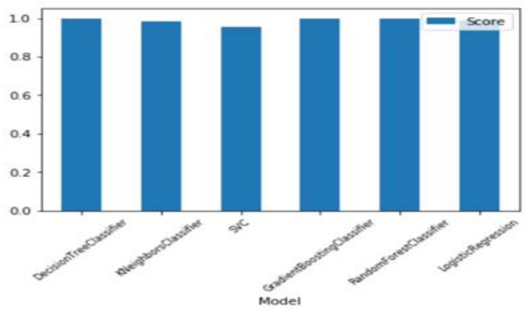


Fig 2.2

From the above Fig 2.1 & Fig 2.2 all the classifiers almost fit the data and there is no **underfitting** visible. This might be due to the reason that classifiers such as gradient boost, decision trees and linear SVM are quite strong in fitting this data as this data is not linearly complex to fit. As the training accuracies of the models were too high for these classifiers we would like to check if there might be a chance of overfitting. We check by analyzing the cross-validation scores of the models by again applying a split on the training data with a **15-fold** cross validation. Fig 2.3 has shown us that some of the classifiers like decision tree, KNN were overfitting the training data based on our cv scores.

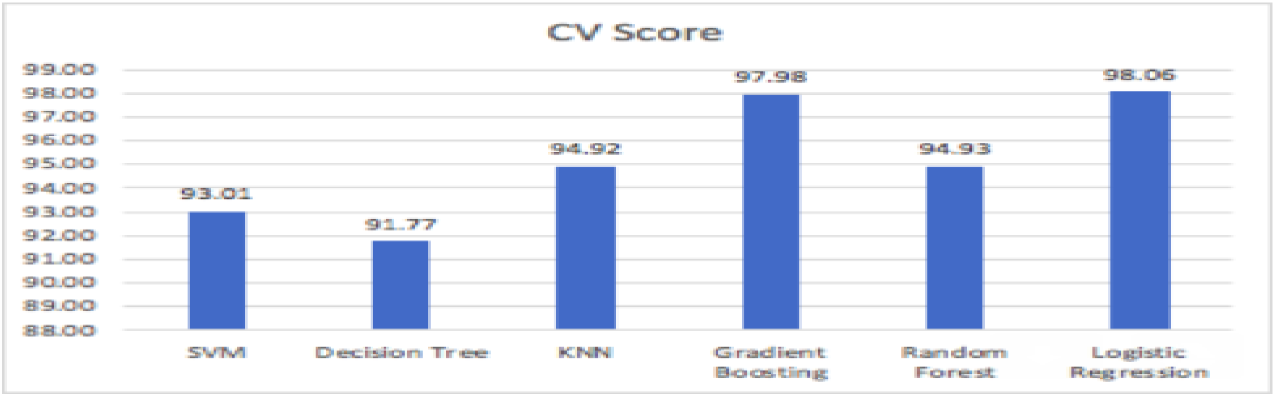


Fig 2.3

Since the cross-validation scores of most models are too close we would like to further check these classifiers efficiency using parameter tuning. We implement **Grid Search** so that optimal parameters for each classifier model could be found. The **GridResearch** basically evaluates all possible combinations of parameter inputs provided and selects the model with best cross validation score. This is a more realistic approach in which **K fold** cross validation measures accuracy of the model based on different combinations of input parameter range.

## Criticizing and Parameter Tuning of Models

### A. Decision Tree Classifier Tuning:

We observed that there was a significant decrease in the cross-validation score of decision tree which implies that it was overfitting the model, still we continued tuning of parameters hoping for a better result. We defined a PARAM grid that contains varied input parameters (criterion, max_features, min_samples_split, min_samples_leaf) with different values. We defined a 10-fold cross validation grid search and obtained the following results:

**{Criterion: 'Gini', max_features: 'auto', min_samples_leaf: 5, min_samples_split: 2}**

We observed a that with the current set of optimal parameters the decision tree model made cross validation accuracy of **91.76%** accuracy. Our decision tree model with tuning gave us a more lower accuracy of **90.1%** which probably states that decision is not able to fit the data perfectly.

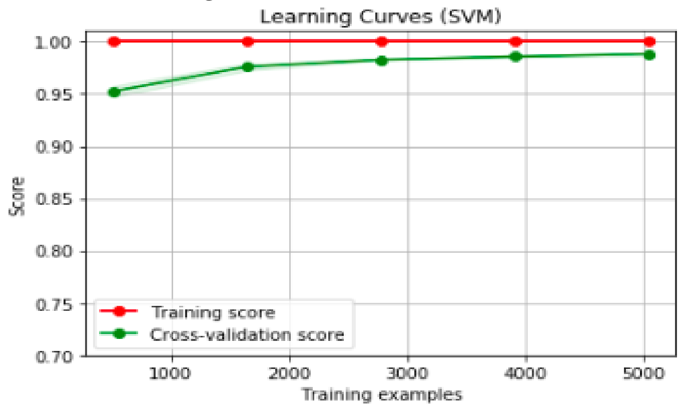
## B. Support Vector Machine (SVM) Classifier: 99% Accuracy

We introduced both linear and rbf kernels in our kernel parameter and varied the values of gamma, ‘C’ over different powers of 10.

Cross Validation Grid Search gave us the following optimal best parameters:

**{Kernel: ‘rbf’, C = 100, gamma = 0.01}**

For the above set of parameters, the SVM classifier obtained an accuracy score of **99%** which shows an improvement of nearly 5% from 93.1% accuracy score of the model with default parameters. It is memory efficient supervised learning classifier as it uses sample of training points to define function. The learning curve for ‘RBF’ kernel classifier is as follows:



## C. Random Forest Classifier: 97.89% Accuracy

We tested Random Forest classifier with default parameter setting which resulted accuracy of 95.95% on test set (70:30 split). After that we tuned the parameters of this classifier using grid search for folds 5 and 10. For both the folds (5 & 10), we kept the range of values for each parameter same which is shown below:

**{ 'n_estimators': [100, 200], 'max_features': ['auto', 'sqrt', 'log2'], 'max_depth' : [1, 5, 10, 15, 20, 25, 30], 'criterion' :['gini', 'entropy'] }**

The parameter tuning using grid search resulted in following optimal values of the parameters:

**With 5 CV:**

**{'criterion': 'entropy', 'max_depth': 15, 'max_features': 'auto', 'n_estimators': 100}**

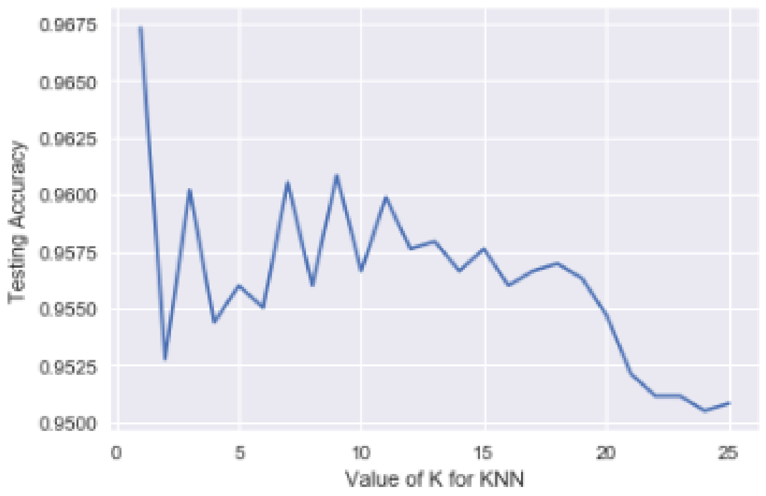
**With 10 CV:**

**{'criterion': 'entropy', 'max_depth': 20, 'max_features': 'auto', 'n_estimators': 200}**

From above parameter values, we were able to get the Random Forest Classifier accuracy score of **97.93%** for 5 folds and **97.89%** for 10 folds.

## D. K-Nearest Neighbor: 96.73% Accuracy

The KNN classifier with default number of neighbors (k=5) gave us the accuracy of 95.6%. Then we tested the classifier for all the values of k ranging from 1 to 25 and recorded the accuracy for each value of k. The graph below shows the variation in testing accuracy of the model based on values of k from 1 to 25:



From the graph above, one can see that the knn classifier shows highest testing accuracy (**96.73%)** for our model.

**E. Logistic Regression: 97.28% Accuracy**

Initially, we applied logistic regression classifier with default parameter setting on our testing dataset (30% of entire dataset) which we obtained through 70:30 split. In this case, our model was trained on the train dataset (70% of entire dataset). With this split we got the testing accuracy (with default parameter setting) of 97.89%. After that we performed the parameter tuning using grid search (10 folds) for parameters 'penalty' and 'C' which resulted in following values of those parameters:

**{ Penalty = 'l2', C = 100}**

With above parameter values, we got the logistic regression classifier accuracy on the test set equal to **97.28%** which is marginally higher than that of in default setting.

**F. Multi-Layer Perceptron: Final Accuracy: 98.12%**

After all the other classifiers, we implemented neural networks to this data as this data is highly dependent on the distance between data points. We started with the default parameter settings and then tuned the following parameters to get the maximum accuracy.

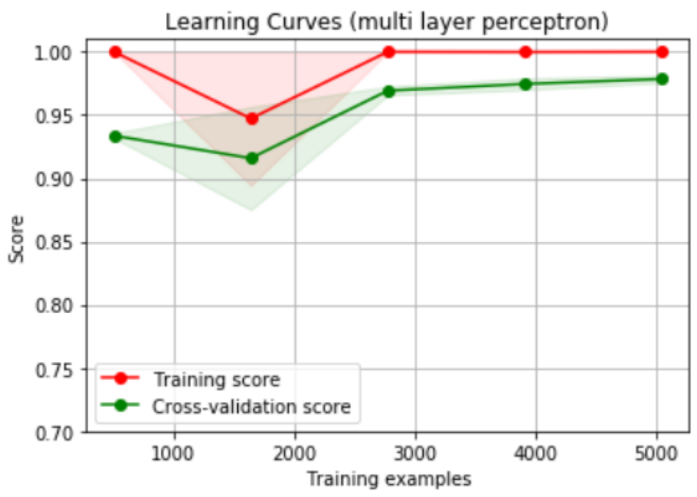
**Hidden_layer_sizes** = 600,800,600

**Activation Function** =Rectified Linear Unit Function

**Optimization Function** = Limited Memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm

**Batch size**=500

**Max iterations** =1000



## Conclusion:

We have concluded that **Support vector machine** classifier with the best parameters(mentioned above) is the best model fit for predicting human activity through sensors. It has an overall accuracy of 99% which is the highest among all the other classifiers. It has handled the non-linearity in the data by applying the 'rbf' kernel to the dataset by transforming it in another space vector.

## Future Work:

Since for time series data, it is highly recommended to use recurrent neural networks for classification, so we propose the use of recurrent neural networks for this data set.
But with our data analysis and model fitting approach via parameter tuning we have almost reached the perfect prediction accuracy. Still no model is a perfect model and we can explore recurrent neural networks for a perfect model in the future.

## References:

**[1]. Want R., Hopper A., Falcao V., Gibbons J.: The Active Badge Location System, ACM Transactions on Information, Systems, Vol. 40, No. 1, pp. 91-102, January 1992**

**[2]. https://en.wikipedia.org/wiki/Activity_recognition**

**[3] https://www.openml.org/d/1478**