ABSTRACT

In the report, Human Activity Recognition system has been presented on the basis of the date gathered by the smartphone sensors. Presently, there are many smartphone applications that already use that data to estimate basic fitness statistics, such as daily step count. But these applications produce a poor method of assessing person's fitness level. This problem has lead people to turn to wearable devices to track their fitness activities. As smartphones are quickly becoming ubiquitous part of the life in the modern world, their embedded sensors have the ability to record a significant amount of data about people's movement.

So in this paper, we are going to test different machine learning approaches to predict the human activity that he/she is performing according to the signals from the cell phone of the individual. Smartphones have inbuilt sensors like accelerometer and gyroscope whose input signals will be recorded and the output of each algorithm is a prediction of the type of activity performed. Finally it is going to recognize six different human activities: Lying, Sitting, Standing, Walking, Walking Downstairs and Walking Upstairs.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

KPCA: Kernelized Principal Components Analysis

PCA: Principal Components Analysis

FFT: Fast Fourier Transform

KNN: K-Nearest-Neighbours

SVM: Support Vector Machine

HAR: Human Activity Recognition

CHAPTER 1

INTRODUCTION

In this fast developing modern world, we are coming across various technologies which are changing our lifestyle. Smartphone is playing a great role in this change. Since recent years, many new different kinds of sensors of much smaller size had been developed which can fit inside smartphones and it had revolutionized the way in which we use them. Even though these recent development in sensors had led into creation of new devices, still our smartphone has enough capabilities to cater majority of our demands.

By using our smartphone and it's sensors, we can easily track various kinds of physical activities done by the user. This information can be used for the well-being of the user. By using machine learning and data science algorithms, one can track the physically activities carried out by human with nearly same accuracy as that of other special purpose devices like fitness bands.

1.1 Applications

It can play a great role in the day to day life of a common human being. As today in this modern world the use of smartphone is growing day by day so this project can be used on a large scale by common person to track day to day activities. It can also reduce the use of external devices like smart watch etc. The tracking of activities can further help the individual to know about how much physical work one needs to do everyday to lead a healthy life. These days people are more focused on electronic gadgets and social media which is also affecting their personal health. So by using this project one can know to keep himself fit and maintain the personal hygiene.

1.2 Motivation

As the size of various electronic sensors is shrinking, more and more different kinds of devices are emerging into the market which is targeted to fulfill specific demands. Wearable devices like fitness band had recently emerged as a health device which senses human physical activities and provides daily status related to the user’s health. The smartphone has nearly same type of sensors which are also available in other fitness devices for tracking human activity. The smartphone had not just become cheap; its market penetration is also very high due to which using it as a device to detect human physical activities can be useful to cater the demands of majority of the people in developed as well as developing countries.

1.3 Objective

Human activity recognition (HAR) is a highly challenging research topic. It aims at determining the activities of a person based on smartphone sensor. The objective of this project is to describe and evaluate a system that uses smartphone based accelerometer and gyroscope to track the user’s physical activity recognition, a task which involves identifying the physical activity a user is performing based on that the person can improve physical activities for better health. It aims in reducing the use of expensive devices like smartwatch.

CHAPTER 2

LITERATURE SURVEY

2.1 Principal Component Analysis

Principal Component Analysis is being used to identify the directions in feature space also called as "Principal Components" along which data varies the most. It uses orthogonal transformation to convert a set of observations of correlated variables into set of values of linearly uncorrelated variables which is called principal component. This transformation is defined in such a way that the first principal component has largest possible variance. It is basically used for the purpose of filtering the data by removing those data which have low or zero variance. Removing those data sets will not affect the results due to low variance.

PCA is basically an orthogonal linear transformation that transforms the data of one coordinate system into another coordinate system. Consider we have data containing m features and n observations and it is given into the form of $m \times n$ matrix, let this matrix be A. Let $\vec{\mu}$ be a vector of length m where μ_i is the mean of all the elements of i^{th} column $\vec{x_i}$.

$$\therefore \; \vec{\mu} = \tfrac{1}{n} (\vec{x_1} + \ldots + \vec{x_n})$$

Let B be $m \times n$ matrix whose i^{th} column is $\vec{x_i} - \vec{\mu}$. Hence $B = [\vec{x_1} - \vec{\mu} \mid \ldots . \mid \vec{x_n} - \vec{\mu}]$. Hence, the covariance matrix S will be

$$S = \tfrac{1}{n-1} BB^T$$

In covariance matrix S, every element S_{ij} gives covariance between i^{th} and j^{th} feature of the given data matrix. Hence S is a symmetric matrix and can be orthogonally diagonalized. Hence, by performing eigendecomposition of the matrix S, we get eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \geq 0$ and eigenvectors $\vec{u_1}, \vec{u_2}, \ldots ., \vec{u_m}$. These eigenvectors are called principal components of data set. In this basis, the largest eigenvalues correspond to the principal components that are associated with most of the covariability among a number of observed data. Hence, the dimensions which are having comparatively small eigenvalues have very negligible impact on the result of the data analysis and hence they can be removed from the matrix S. This helps in reducing the dimensions of the data to be processed and hence reduces the noise of the data.

2.2 Kernelized Principal Component Analysis

Kernelized Principal Components Analysis (kPCA) is an extension of Principal Component Analysis. When the given data of n observations and m features are not linearly separable, then it is necessary to map those observations into N dimensions where $N > m$. Kernelized Principal Component Analysis (kPCA) attempts to identify similar variance-maximizing directions, but does so after the data have been mapped the generated covariance matrix of m dimension into some alternate space, via a feature mapping function of dimensions N. For eg. Consider the data points plotted into a 2-D feature space as shown below

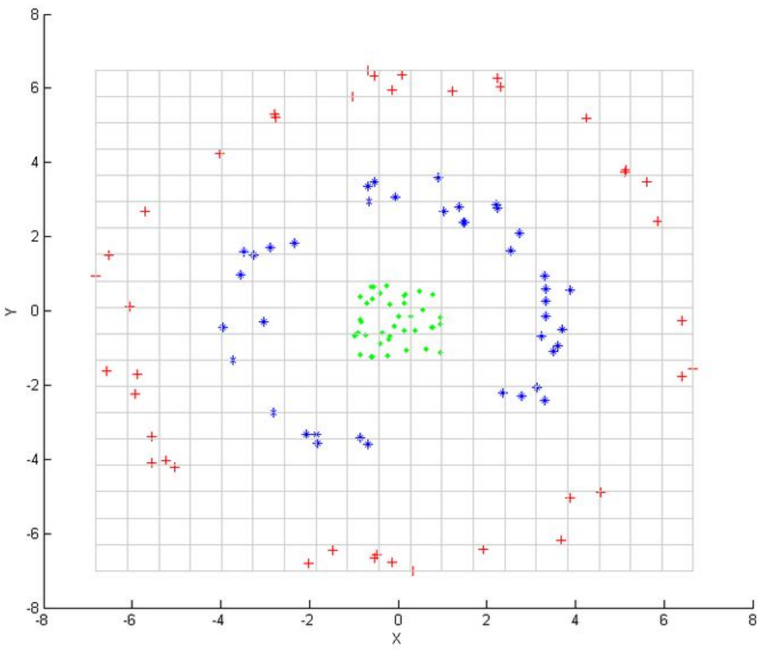

Fig. 1. kPCA Input

It is very clear that there is no method to separate the red and blue points using through a single line. Hence the data points of features x_1, x_2 need to be mapped into another space of higher dimension.

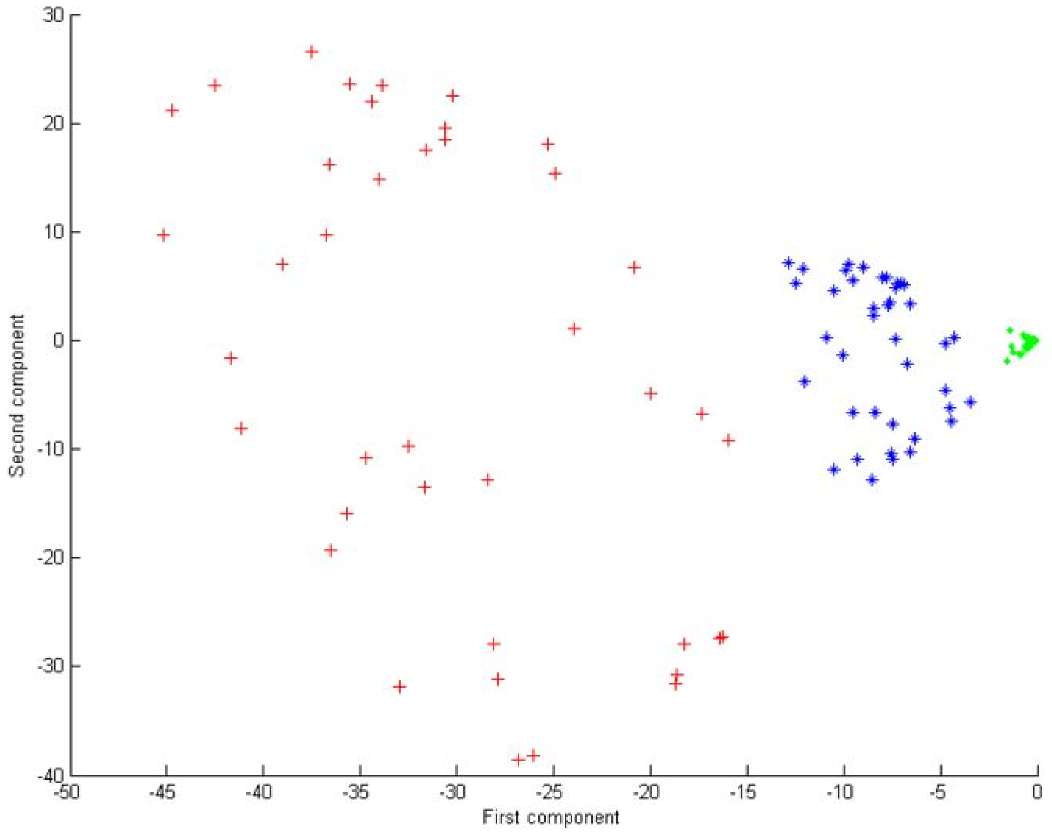

Fig. 2. kPCA output

Let ϕ be the function which maps the point into a space of dimension N.

$$\therefore \phi : R^m \to R^N;\ x \to X$$

Hence, covariance matrix $C_F = \frac{1}{n}\sum_{j=1}^{n} \phi(x_j)\phi(x_j)^T$. In covariance matrix C_F, $\sum_{i=1}^{n} \phi(x_i) = 0$. Hence, eigenvector $v = \sum_{i=1}^{n} \alpha_i \phi(x^i)$ where $\vec{\alpha} = [\alpha]^n_{i=1}$ is a normalized eigenvector of matrix $K = [\phi(x^{(i)})^T \phi(x^{(j)})]_{i,j=1}{}^m = [K(x^{(i)}, x^{(j)})]_{i,j=1}{}^m$. Here

$K(x,y)$ is the kernel function associated with ϕ. Hence, score of an example x of the j^{th} kPC gives the projection $v_j{}^T \phi(x) = \sum_{i=1}^{n} \alpha^j{}_i \phi(x^{(i)})^T \phi(x) = \sum_{i=1}^{n} \alpha^j{}_i K(x^{(i)}, x)$. Thus, any kPC score can be computed without explicitly representing $\phi(x)$. Because ϕ can map the data to a high dimensional space, kPCA can often capture nonlinear relationships among the data points, which are missed by the standard, linear version of PCA, described previously.

2.3 k-Nearest-Neighbours

k-Nearest-Neighbours (kNN) algorithm classifies a point based on the points which are nearest to it. The predicted probability if a point falls into class j is proportional to the number of the k training points nearest to desired point which are in class j. Mathematically, the probability of a desired point to fall in class j is given as:

$$P(y^{(i)} = j | x = x^{(i)}) = \frac{1}{k} \sum_{q \in N} 1\{y^{(q)} = j\}$$

where, N - set of indices of the k points closest to $x^{(i)}$

Thus a desired point is predicted to belong to the same class as like the other nearest k training points. Further validation set is used to select the number of principal components or kernelized principal components and also the optimal value of k in kNN.

2.4 Support Vector Machine

Support Vector Machines are the binary classifiers which attempt to find a vector of parameters α to minimize the regularized loss function

$$J(\alpha)_\lambda = \frac{1}{m} \sum_{i=1}^{m} [1 - y^{(i)} K^{(i)^T} \alpha] + \frac{\lambda}{2} \alpha^T K \alpha$$

where, K = $[K(x^{(i)}, x^{(j)})]_{i,j=1}^{m}$ = $[K^{(1)} \ldots K^{(m)}]$ $\exists$ kernel function $K(x,y) = x^T y$.

As Support Vector Machines work well for binary classifications, it can also be used in many-class classification problems in two ways:

1. One vs One

In this method, the $\binom{L}{2}$ SVMs are trained in such a way that each SVM serves to separate one pair of classes from each other. The desired point is predicted to be in the class that is chosen most frequently, when the point is evaluated using all of the SVMs.

2. One vs All

In this method, the L SVMs are trained in such a way that each SVM serves to separate one class from rest of the classes. The desired point is predicted to be in the class for which its signed distance to the decision boundary is greatest.

CHAPTER 3

DATA PREPROCESSING

Many smartphone applications already use that data to estimate basic fitness statistics, such as daily step count. However, the simplistic metrics that these applications produce are a poor method of assessing a person's activity level. As a result, many people have turned to wearable devices to track their fitness activities.Our aim of this experiment is to achieve maximum accuracy on the simple data produced by smartphone sensors.

3.1 Data

The data set on which we test our methods is courtesy of the UCI Machine Learning Repository. It contains summaries of accelerometer and gyroscope readings from waist-worn cell phones by 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.The data contain 10,299 observations with about 300-400 observations per subject, and a roughly even distribution of observations across activities.

3.2 About Data In-depth

Following is the in-depth information about the features within the dataset related to various sensors

1. **Body Acceleration:** The features related to body acceleration can be classified into two categories on the basis of the domain i.e. time domain and frequency domain. The data generated by the accelerometer gives us acceleration signals in time domain. In order to generate signals under frequency domain, fast fourier transform (FFT) is applied over the generated sample windows. Along with it the jerk signals were computed by differentiating the acceleration signals over the time domain. Now, on each sample signal, various statistical functions like mean, standard deviation, interquartile range, mean absolute difference, etc. were calculated along X, Y, and Z axis in the physical space as well as on the magnitude of the acceleration vector.
2. **Gravity:** The signals related to gravitational acceleration were gained by applying a Butterworth low-pass filter on the signals of accelerometer with a cutoff frequency of 0.3Hz. This also removes noise from the sample data generated by sensors as value of g(gravitational acceleration) theoretically never change by magnitude, but they change by the direction in which the accelerometer had sensed it due to the change in position of the smartphone in space. Just like body acceleration, various statistical functions were applied on the gravitational acceleration signals along time and frequency domain. As for gravitational acceleration, jerk signals were not computed.
3. **Gyroscopic signals:** Gyroscope generates angular velocity and acceleration signals along time domain which varies along X, Y, and Z axis in the physical space due to motion of the subject (person) in real time. FFT is applied on the generated sample signals to generate frequency domain signals as well as differentiation is applied to generate jerk signals.Various statistical functions were also applied over it.

3.3 Feature Engineering

Feature engineering is performed in order to find necessary dimensions i.e. features which have greater contribution in classifying data points and remove those features which have very less or no impact over it. This reduces the numbers of features given as input to the classifier as certain classifiers takes huge amount of time on classifying high dimensional dataset. Following are the 2 feature engineering algorithms used by us in this experiment

1. **Principal Components Analysis (PCA)**: Figure 3 shows the amount of variance preserved w.r.t no. of components retained after applying PCA. As it can be seen from the graph, the amount of variance explained saturates after taking 100 components. Hence, it is more likely that the components above 100 will have very less impact in the classification process.

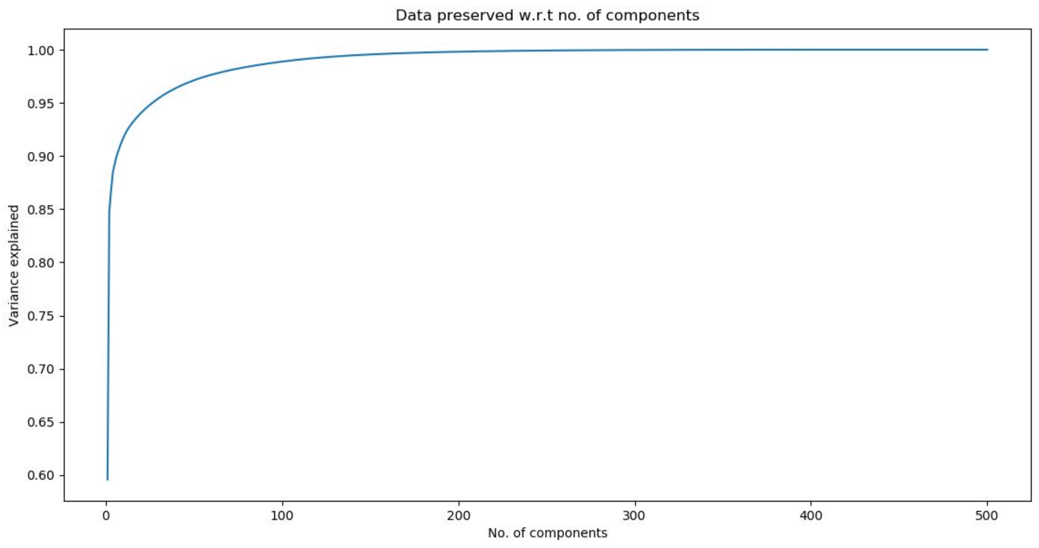

Fig. 3. Cumulative Variance V/S No. of components

After applying PCA, we had plotted few components in 3D space in order to analyse the weather the data points are separable or not. From figure 4, it can be observed that static motions i.e. sitting, standing and lying can be easily separated from dynamic motions i.e. walking,

walking upstairs and walking downstairs. Also in plot of figure 5, all the 3 static motions can be seen separate.

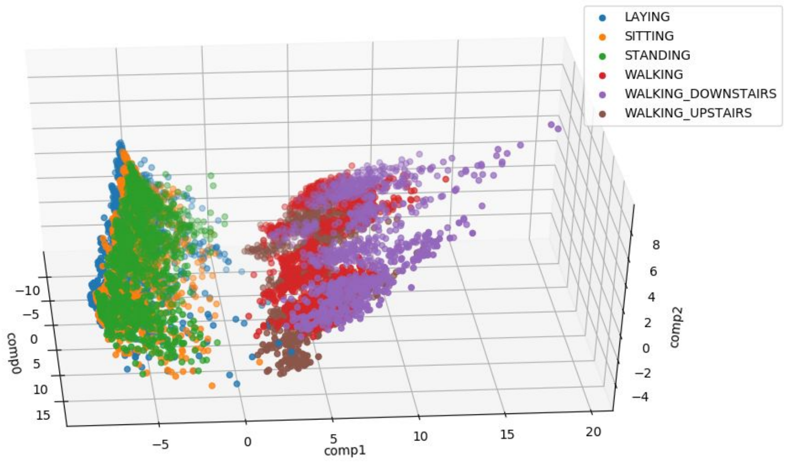

Fig. 4. 3D-plot of first 3 components of PCA

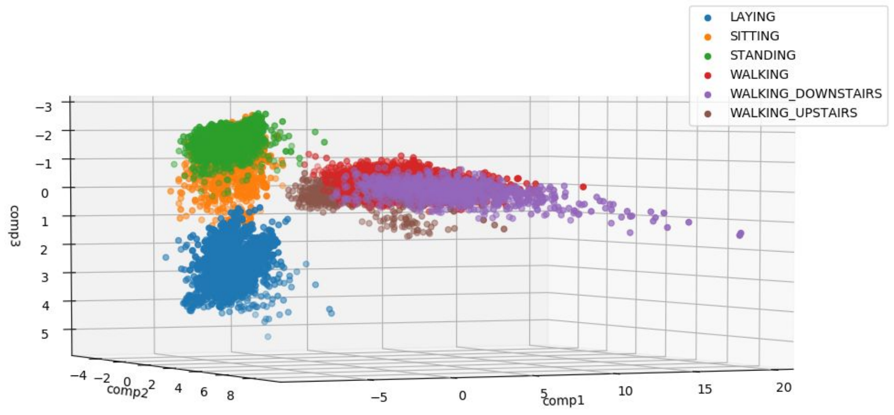

Fig. 5. 3D-plot of 2nd, 3rd, and 4th components of PCA

2. **Kernelized Principal Component Analysis (kPCA):** The figure below shows the eigenvalues of each component after applying kPCA. It can be seen that the eigen values drop drastically just after the few 5-10 components.

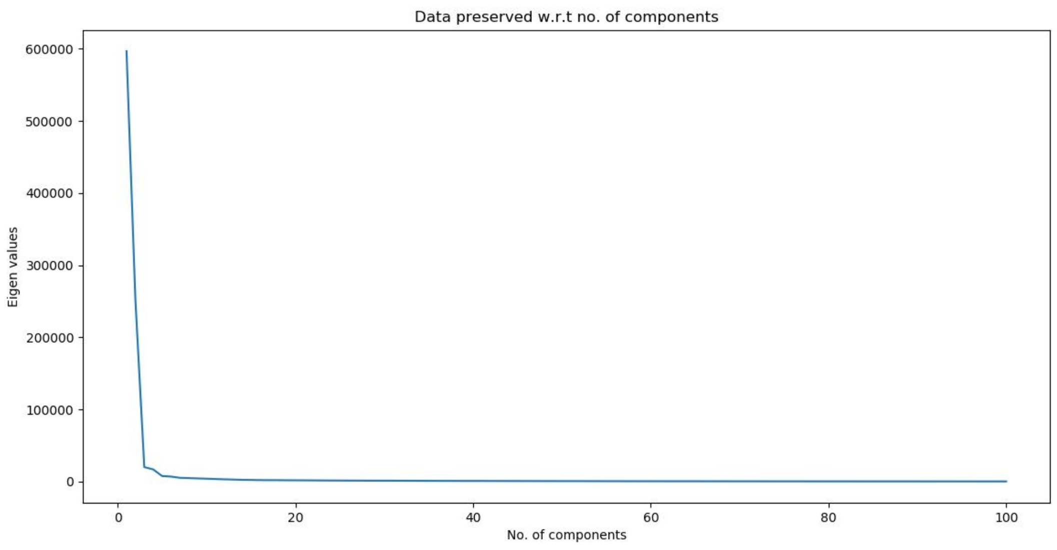

Fig. 6. Eigenvalue V/S No. of components in kPCA

Figure 7 shows clear separation of static motions and dynamic motions.

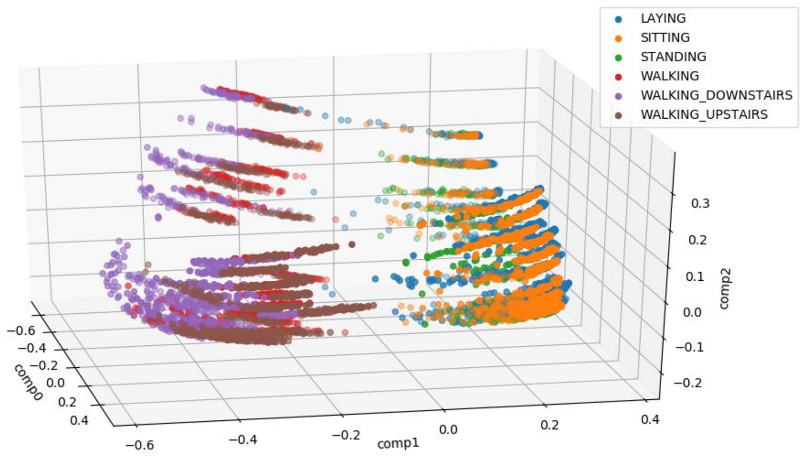

Fig. 7. 3D-plot of first 3 components of kPCA

CHAPTER 4

CLASSIFICATION

After successfully applying PCA and kPCA on the dataset, we used various machine learning classification algorithms to check how they work on datasets and check about the in depth analysis for their prediction.Various classification algorithms used in this project are as follows:

4.1 K Nearest Neighbour

We tried to implement KNN algorithm in different ways. First of all we implemented it on the training dataset without applying PCA. We then calculated the training accuracy, testing accuracy and cross validation accuracy for the datasets. In the following graph we can clearly see about how all these accuracies vary depending upon different values of k.

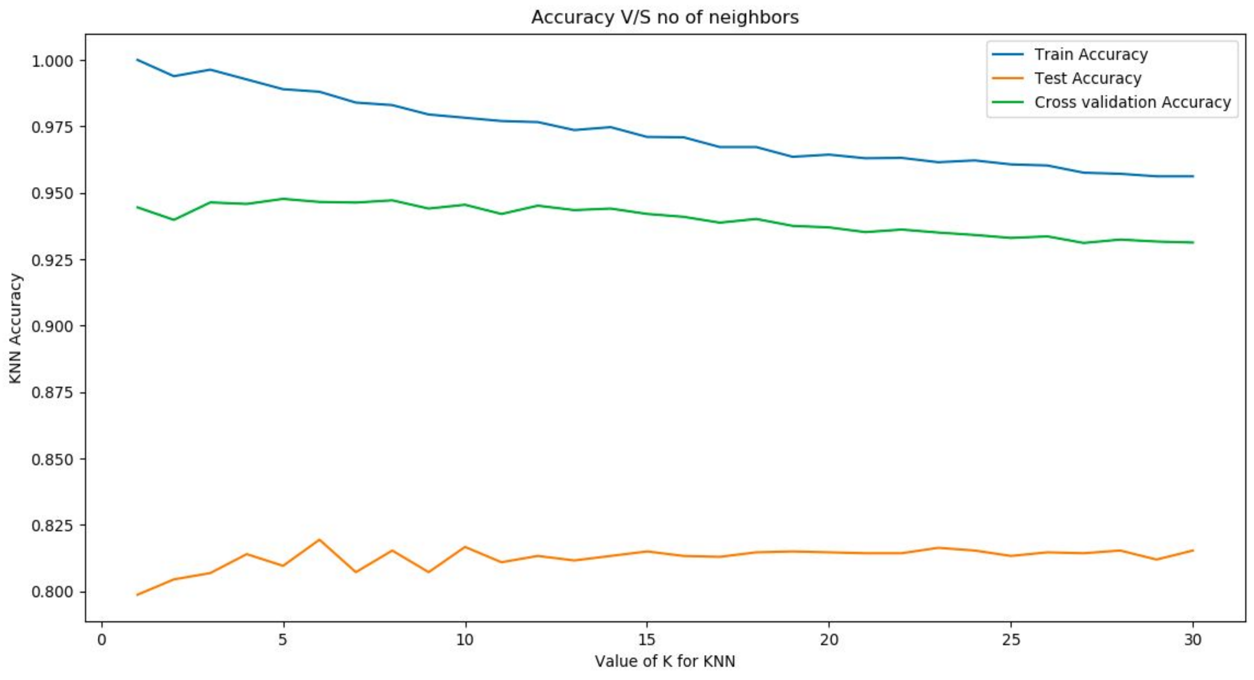

Fig. 8. Accuracy of KNN vs no of nearest neighbours

There was not much change found in the graph before and applying pca so only single graph has been shown above. So after observing both of the above graphs, we finally came up to a conclusion that we would be using the value of **n_neighbors = 10**. We also tuned different parameters of the KNN to get the maximum accuracy and thus finally came up with the accuracy of **81.67 %**. After finding the classification accuracy, we further draw the confusion matrix for the same and came up with the following confusion matrix:

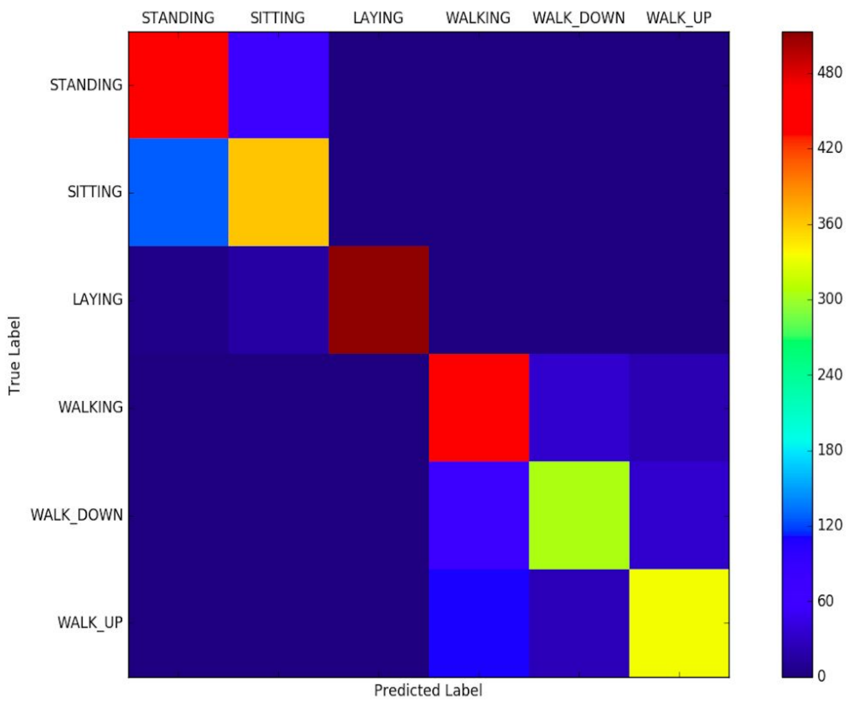

Fig. 9. Confusion Matrix for KNN algorithm

In the above confusion matrix, we can clearly see KNN algorithm does not distinguishes properly between walking, walking upstairs and walking downstairs. Also it faces problems in distinguishing sitting and standing. Even and tuning different parameters and changing the value of k, accuracy increased from nearly **79.32%** to **81.67%** . In the next page, a table is shown which shows about the details of confusion matrix detailing about the number of actual value vs the predicted value for each and every movements. This can further help us to calculate the precision and recall values which will be helpful in calculating the null accuracy. Null accuracy is also important since classification accuracy does not gives the idea about how predicted results are behaving. This table gives us a detail view of the area of concern for our algorithm where further improvements can be done to improve accuracy.

	Predicted						
Actual		Standing	Sitting	Laying	Walking	Walk Down	Walk Up
	Standing	457	74	0	1	0	0
	Sitting	128	361	0	0	0	2
	Laying	0	17	513	0	0	1
	Walking	0	0	0	437	35	24
	Walk Down	0	0	0	79	305	36
	Walk Up	0	0	0	112	25	334

Table. 1 Confusion matrix obtained by KNN on test dataset

4.2 Random Forest Classifier

Now after applying KNN algorithm, we tested our datasets on Random Forest Classifier algorithm and found impressively very good results. We applied this algorithm on both datasets i.e, datasets after applying pca and datasets without applying pca. There was a difference of nearly 1.5% between both of them. Further we applied this algorithm to get both training as well as testing accuracy. And finally came up with a final result of **n_estimators = 200** giving classification accuracy of **90.26%**. While calculating the accuracy various other parameters were also tuned. The best parameters which contributed to this accuracy were n_jobs=4 and min_samples_leaf=10. Below is the graph that we obtained while calculating testing and testing accuracies. It may seem that there is a difference of 1.5% among the accuracies but training the model with pca components is quite effective since it reduces overall cost and time and is much efficient than the other.

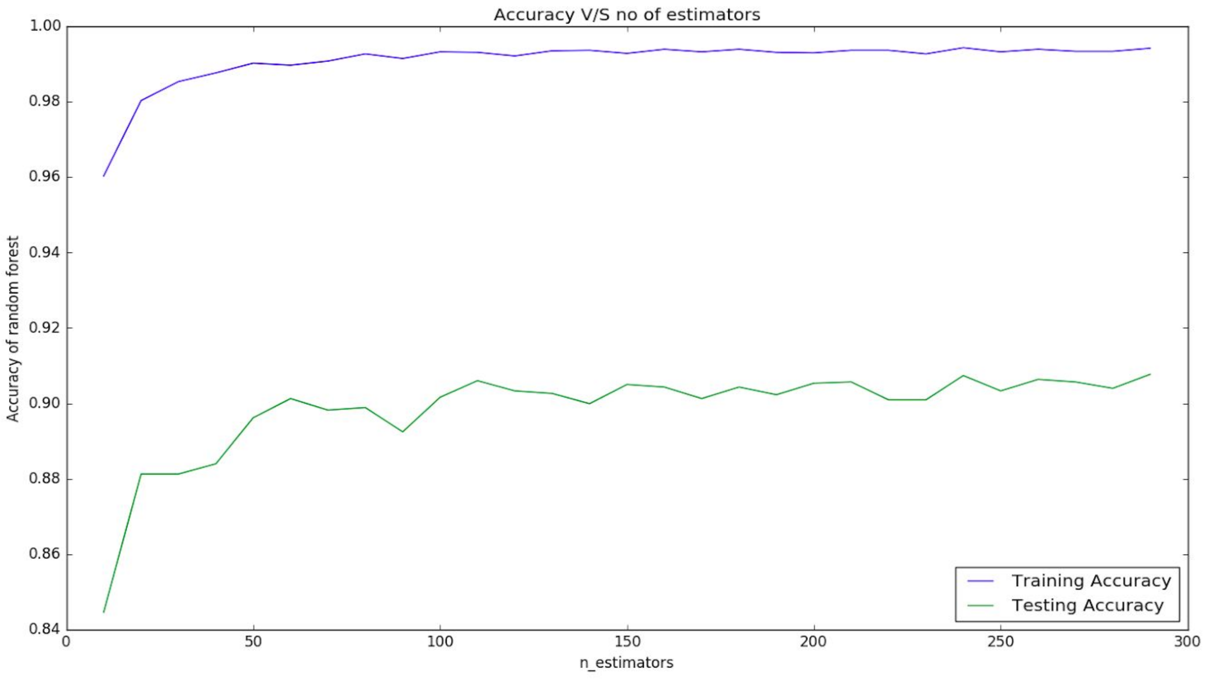

Fig. 10. Accuracy of Random Forest Classifier Vs n_estimators(On PCA components)

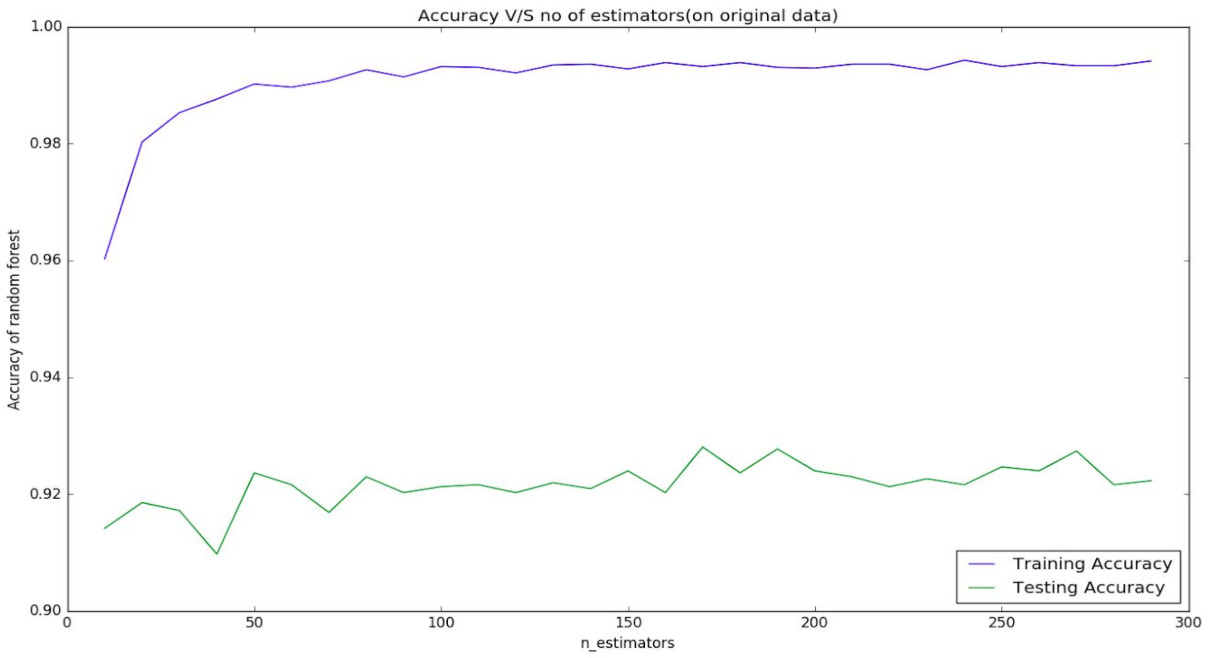

Fig. 11. Accuracy of Random Forest Classifier Vs n_estimators(On original datasets)

So after checking the classification accuracy of the algorithm, we went to find the confusion matrix about the algorithm to get the in depth knowledge of how our algorithm is working on the datasets.

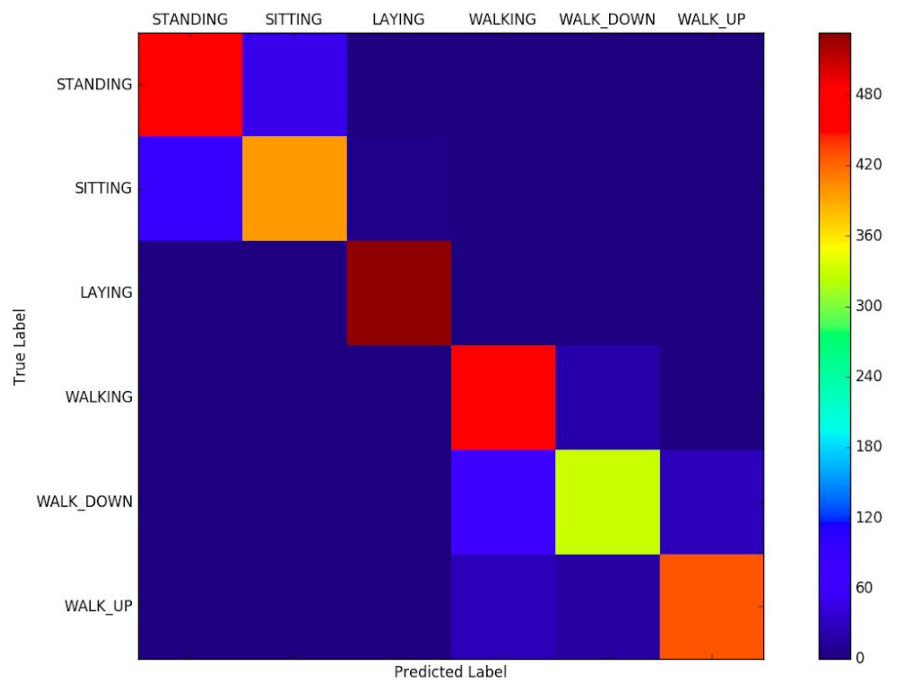

Fig. 12. Confusion Matrix for Random Forest Classifier

		Predicted					
Actual		Standing	Sitting	Laying	Walkin g	Walk Down	Walk Up
	Standing	483	49	0	0	0	0
	Sitting	90	396	5	0	0	0
	Laying	1	2	533	0	0	0
	Walking	0	0	0	475	20	1
	Walk Down	0	0	0	61	330	29
	Walk Up	0	0	0	27	17	427

Table. 2. Confusion matrix obtained by Random Forest Classifier on test dataset

Now from the above confusion matrix we can see that how Random Forest Classifier predicts more accurately on the datasets as compared to KNN algorithm which contributed towards in the increase in accuracy of the same. However it is still facing the problem of standing and sitting classification and little bit in walking problems.

4.3 Support Vector Machine(SVM)

As shown in the figure 12, by including more PCA components, the training and testing accuracy increases drastically, after which it began to saturate. According to the series of tests ran by us, we had got a stable and decent accuracy by taking first 200 components generated by PCA.

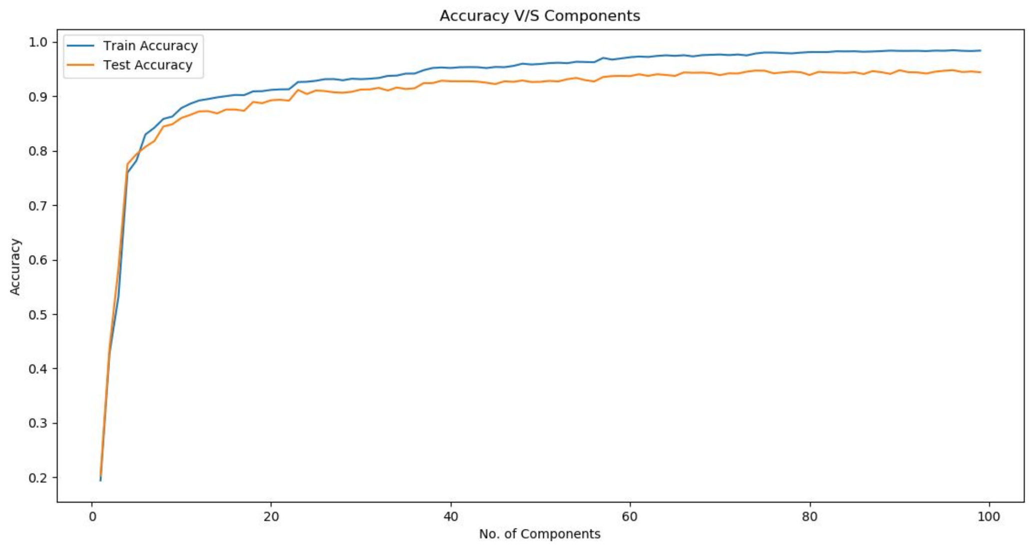

Fig. 13. Accuracy V/S No. of PCA Components

To obtain optimal classification model from SVM, we had used linear kernel i.e. the SVM with linear hyperplane. The accuracy achieved by using 200 PCA components is 96.4% over testing data with 99.5% accuracy on training dataset. The results for both one V/S one and one V/s all were same with very little or no variations. Figure 13 shows the plot of its confusion matrix and Table 2 shows the confusion matrix.

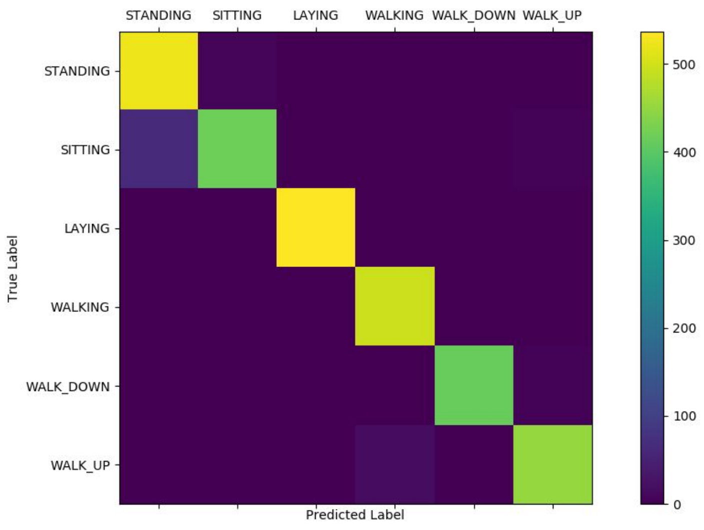

Fig. 14. Confusion matrix plot in linear SVM

	Predicted						
Actual		Standing	Sitting	Laying	Walkin g	Walk Down	Walk Up
	Standing	523	8	0	1	0	0
	Sitting	66	419	1	0	0	5
	Laying	0	0	537	0	0	0
	Walking	0	0	0	496	0	0
	Walk Down	0	0	0	2	413	5
	Walk Up	0	0	0	18	0	453

Table 3. Confusion matrix obtained by Linear SVM on test dataset