Faculty of Sciences

Sensor Data Analysis

Omkar Kulkarni

Master dissertation submitted to
obtain the degree of
Master of Statistical Data Analysis

Promotor: Prof. Dr. Christophe Ley
Co-promotor: Ravindar Roopreddy, CloudLeaf Inc

Department of Applied Mathematics, Computer Science and Statistics

Academic year 2016 - 2017

Omkar Kulkarni
June 24, 2017

Foreword

This diploma thesis is entirely the result of my own work, I mentioned all the sources of information I used. This thesis was carried out as a project for CloudLeaf Inc [1]

Data was obtained under the terms of The Biomedical Informatics & eHealth Laboratory (BMI Lab), which resides at the Dept. of Informatics Engineering of the Technological Educational Institute of Crete (TEI Crete) [11]. The dataset is available for research and educational purposes. For the streaming, data was self generated.

First of all, I would like to express my immense gratitude to my promoters Dr. Christophe Ley and Co-promoter from CloudLeaf Mr. Ravindra Roopreddy, for their support guidance and encouragement. Without Mr. Roopreddy's motivation I would never have used spark streaming. I am very grateful for the meetings I had with Dr. Ley: each of them used to give me a burst of energy for doing things better and not deviate from the fundamentals. I would also like to thank all the contributors to apache spark, R, Python without whom such amazing platforms would not be available as open source.

My sincere thanks also goes to Mr. Raj Kulkarni for offering me the internship opportunity at CloudLeaf Inc and leading me to work on a diverse and exciting project. I would like to give my deepest thanks to my brother Shailesh and Mithila for all the support and encouragement during the whole process.

[1] For more information contact CloudLeaf Inc http://www.cloudleaf.com/

Table of Contents

Abstract

Older people are a rapidly growing proportion of the world's population is reported by National Institute on Aging in 2015. Elderly people have a higher risk of death or injury resulting from falls. A high proportion of over-aged population fall each year which can result into serious health challenges. The increase in population of elderly people also increases the demand for health care systems focused of elderly people.

The use of smartphones to detect fall events is a very popular method due to the decrease in the manufacturing costs of smartphones as a result of the development of inexpensive Micro Electro Mechanical System (MEMS) sensors. These sensors provide better computational capabilities when compared to other wearable devices. Smartphones generally include the following sensors: accelerometer, gyroscope, compass, magnetic field,proximity light and pressure sensors. This allows different algorithms to be applied to increase the accuracy of any data required.

In this thesis, the recognition of and the differentiation between fall activities and activities of daily living (ADL) was performed using a publicly available dataset, MobiAct and MobiFall. The data set has captured inertial measurement unit (IMU) values for different ADL and Fall activities. A large database was constructed to train and validate the model. For sensor, using windowing technique the time domain and frequency domain (Discrete Fourier Transform) features were extracted across each axis of motion. Feature selection methods were implemented to reduce dimensionality, principal component analysis was used for exploration. Different classification algorithms like k-NN, decision trees, SVM were implemented and evaluated based on their accuracy, sensitivity, and specificity to decide on the final model.

Furthermore, for the implementation of real time fall detection system we used Apache Spark Streaming framework via pyspark.streaming module. Using basic TCP socket IMU signals were sent to spark framework as data stream and using moving window the features were extracted for classification into one of the activities defined in the MobiAct dataset.

1 Introduction

The research for this thesis was carried out as a part of the CloudLeaf Inc project. It was started with the goal of using Apache Spark Streaming, to deliver outputs in real time. The case of phone fall detection was chosen. The problem of detecting elderly fall is indeed an old problem. This work is motivated to explore real time classification using machine learning algorithms and implement them so as to give real time classification with minimum latency, and hence try to address the detection of fall. Moreover, detection of activities of daily living (ADL)s along with falls, is also carried out.

Fall detection systems are categorized into the following three groups: ambience device, camera-based systems and wearable devices [4][7]. All the systems have their advantages and disadvantages. For instance, Ambience devices are attached around an area which can detect falls using the following sensors: pressure, PIR, Doppler, microphone and accelerometer sensors. The advantages of this method is that it is cheap and non-intrusive however, disadvantages include the range and environmental factors which can result in low accuracy [4].Computer vision makes use of cameras to track the user movements. Advantages of the system is that it can detect multiple events simultaneously and are less intrusive since they are not worn [4]. The disadvantages of this system are that it is limited to a specific area and does not guarantee privacy and is expensive [7], [4].

We focus on using a system of sensors available on a smart phone to detect falls and ADLS. Moreover, there is a steep decrease in cost of smart phones and also increase in variety of sensors

placed on smart phones with inexpensive Micro Electro Mechanical System (MEMS)sensors [9]. It is the technology of microscopic devices, particularly those with moving parts. It merges at the nano-scale into nanoelectromechanical systems (NEMS) and nanotechnology. These sensors provide better computational capabilities when compared to other wearable devices [4]. Most of the smart phones include gyroscopes, accelerometers, compass, magnetic field sensors, pressure sensors, proximity light sensors. This allows for various kinds of feature extraction to be used in algorithms.

The static data was collected from MobiAct Dataset [12] and for streaming data a Moto G2 smart phone was used. Static data mainly consisted of time series data of accelerometer, gyroscope and orientation. These three sensors are basic units in any Inertial navigation system, as the tri-axis accelerometer captures the acceleration acting on the smart phone in all three axis also capturing acceleration due to gravity, similarly three axis gyroscope captures the angular velocity of the phone in all three axis of rotation and orientation is internally calculated. Using windowing techniques, various time domain and frequency domain statistical features were extracted. Feature selection procedure was carried out using random forest variable importance selection. Our approach to the multi class classification was based on support vector machines with *radial* kernel. Other classification algorithms were tested such as Random Forest and KNN, SVM with polynomial and linear kernels. The process of feature selection, model building was done in R version 3.4.0, (*nickname : You Stupid Darkness*)

The Radial kernel SVM was implemented in the spark streaming framework and it was trained using the static data to classify the streaming data. Statistical features were extracted from the streaming data using sliding window and used in the classifier. For spark streaming Python 3.5.2 platform was used with Spark 2.1.0 and PySpark API.

There was a significant gap between the static data and real life streaming data settings. However, its results are promising and can serve as a solid ground for further research.

2

Data and methods

2.1 Data collection

2.1.1 Static Data Collection

The data to classify different ADLs and Falls of the every day life was used from the MobiAct data set. The aim was to use a versatile dataset that enables the objective assessment of fall detection and even activity recognition methods. The dataset is called MobiAct as its previous version was MobiFall primarily focusing on fall detection. ”MobiAct” is based on the similar lines of ”MobiFall” dataset and is the next version of it [12].

Data from the accelerometer and gyroscope sensors (plus orientation data; the orientation sensor is software-based and derives its data from the accelerometer and the geomagnetic field sensor) of a smartphone were recorded 2.1 . Specifically, a Samsung Galaxy S3 device with the LSM330DLC inertial module (3D accelerometer and gyroscope) was used to capture the motion data. The gyroscope was calibrated prior to the recordings using the devices integrated

Type	Values	Description
accelerometer	x,y,z(m/sˆ2)	Acceleration force along the x y z axes (including gravity).
gyroscope	x,y,z(rad/s)	Rate of rotation around the x y z axes (Angular velocity).
orientation	Azimuth,Pitch,Roll	Angle around the z x y axes. (degrees)

Table 2.1: Sensor Description. Data from 3 sensors were captured in the process. Accelerometer at around 100 Hz, and gyroscope and orientation at around 200 Hz.

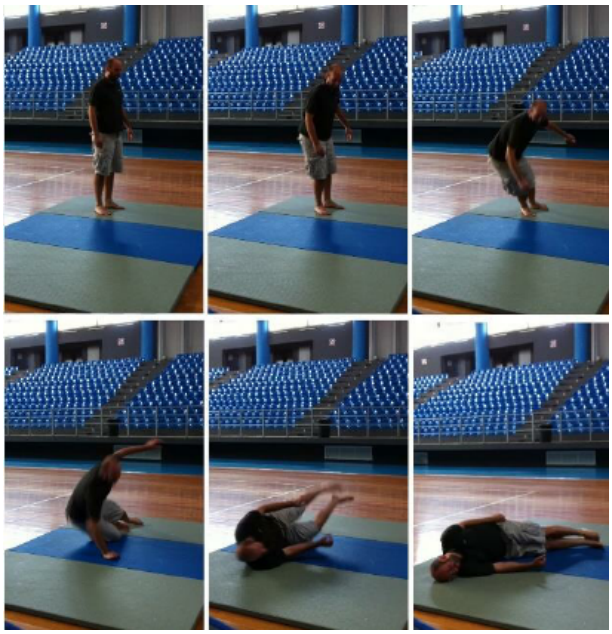

Figure 2.1: Illustration : A subject/volunteer simulating a 'Fall' during data collection process for MobiAct Data Set. Courtesy BMI Lab.

tool. Dataset has raw data for acceleration, angular velocity and orientation with the enabled parameter SENSOR_DELAY_FASTEST. This provides the highest possible sampling rate. The signals can be subsampled at any time if lower sampling rates are desired.

In an attempt to simulate every-day usage of mobile phones, the device was located in a trouser pocket freely chosen by the subject in any random orientation. For the falls, the subjects used the pocket on the opposite side of the falling direction.For the fall simulation a relative hard mattress of 5 cm in thickness was utilized to dampen the fall. The ADLs were chosen based on their commonness and on their similarity to actual falls, *which may produce false positives*.

Dataset encompasses four different types of falls 2.3 and nine different ADLs from a total of 57 subjects with more than 2500 trials, all captured with a smartphone. It had 42 males and 15 females(age: 20-47 years, height: 1.60-1.93 m, weight: 67-120 kg).

The activities of daily living were selected based on the following criteria 2.2 : a) Activities which are fall-like were firstly included. These include sequences where the subject usually stays motionless at the end, in different positions, such as sitting on a chair or stepping in and out of a car; b) Activities which are sudden or rapid and are similar to falls, like jumping and jogging; c) The most common everyday activities like walking, standing, ascending and descending stairs (stairs up and stairs down).

2.1.2 Real Time Streaming Data

To generate data in streaming manner a *Android Version 6.0* phone device - Moto G(2nd Generation) was used. The device was installed with an open source application "SensorStreamer"

id	Code	Activity	Trials	Duration	Description
1	STD	Standing	1	5m	Standing with subtle movements
2	WAL	Walking	1	5m	Normal walking
3	JOG	Jogging	3	30s	Jogging
4	JUM	Jumping	3	30s	Continuous jumping
5	STU	Stairs up	6	10s	Stairs up (10 stairs)
6	STN	Stairs down	6	10s	Stairs down (10 stairs)
7	SCH	Sit chair	6	6s	Sitting on a chair
8	CSI	Car-step in	6	6s	Step in a car
9	CSO	Car-step out	6	6s	Step out a car

Table 2.2: Activities of Daily Living

id	Code	Activity	Trials	Duration	Description
10	FOL	Forward-lying	3	10s	Fall Forward from standing
11	FKL	Front-knees-lying	3	10s	Fall forward from standing
12	BSC	Back-sitting-chair	3	10s	Fall backward while trying to sit on a chair
13	SDL	Sideward-lying	3	10s	Fall sidewards from standing

Table 2.3: Falls

[1]

This application captures sensor data in real time and streams it over Transmission Control Protocol and Port (TCP:PORT) to the target device, in the form of JSON objects, in this case a system with Spark Streaming 3.1 framework to process this stream of data.

The sensor module on the phone is BOSCH-BMC150 sensor module. It is a micro-electro-mechanical-system of sensors (MEMS), it is comprised of 3-axis Accelerometer unit, 3-axis Gyroscope unit, 3-axis magnetometer unit.

2.1.3 Accelerometer

Accelerometers detect the acceleration due to external forces acting on objects. The output of an accelerometer is combination of the acceleration due to the Earths gravity and the linear acceleration due to motion [2]. Due to MEMS-micro electro mechanical systems, the cost of installing accelerometer on silicon in our case a mobile phone is extremely low. Hence, using sensors from phones in a way turns to be cost effective. We have a system of three axis viz X,Y,Z accelerometer. The acceleration associated to external forces acting on an object can be explained by Newtons second law of motion $F = ma$, where a is the acceleration and m represents the mass.

[1] Github Repository `github.com/yaqwsx/SensorStreamer`

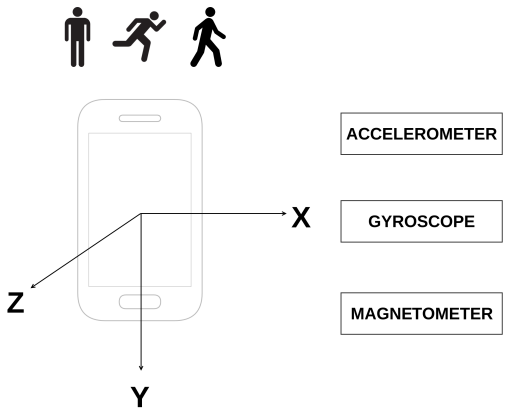

Figure 2.2: Collect Accelerometer, Gyroscope, Magnetometer data at 100 Hz, for actions Walk, Run, Stand, Fall etc

2.1.4 Gyroscope

Gyroscopes give us the 'angular rate' (given the axis of measurement) which is the speed the angle is changing. If ϕ be the angle(in radians), then

$$gyroscope\ output = \omega = \frac{d\phi}{dt} \tag{2.1}$$

Where ω is the angular rate. MEMS gyroscopes work based on the Coriolis effect.These sensors measure the Coriolis acceleration acting on a vibrating element which is used as proof mass [9] . The vibrating elements are available in the form of a tuning fork or a vibrating wheel. All MEMS gyroscopes are rate gyroscopes. MEMS gyroscopes have many advantages compared to mechanical and optical gyroscopes. MEMS gyroscopes have low power consumption, small size, low weight, hight reliability, low maintenance cost, fast start up and lower price.

2.1.5 Orientation

To calculate the orientation of the device, also known as Yaw, Pitch, Roll, one can derive it from accelerometer outputs or by integrating gyroscope. Ideally to get the angle ϕ, we need to integrate the $angularrate$ over time.

$$\phi = \int \omega dt = \sum \omega dt \tag{2.2}$$

But with MEMS gyroscope, the ω is associated with noise/error, and with the integration this noise too gets added! Hence, we observe a drift in the angle ϕ eventually. So usually, some form of $Sensor\ Fusion$ is performed fusing accelerometer and gyroscope values (sometimes

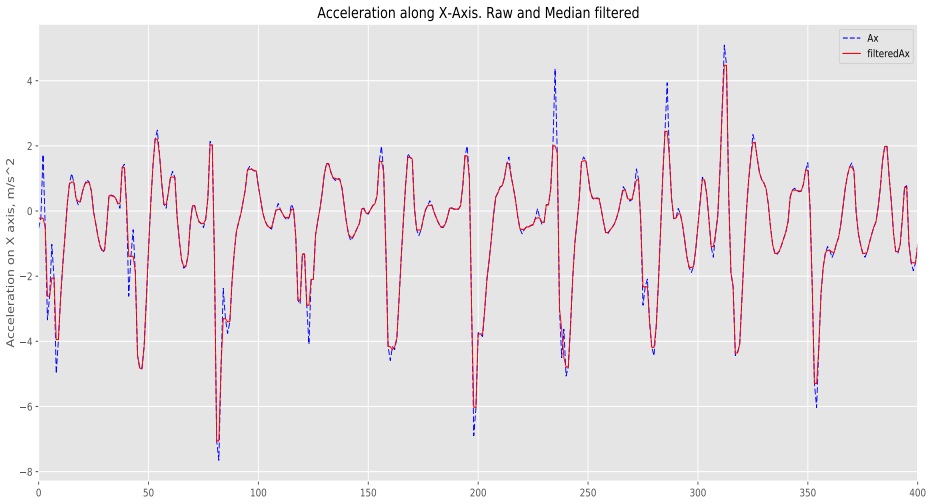

Figure 2.3: Representation of 3rd order median filter on X-axis accelerometer value to depict trimming of leaks.

also magnetometer values) to get more reliable values. Fortunately, the Android system (also iOS systems) provide with calculation of orientation internally [2] [3].

2.2 Feature Extraction

2.2.1 Data Pre-Processing

Before feature extraction, the accelerometer data is passed through 3rd order median filter implemented in Python SciPy [1]. It helps to attenuate the noise produced by External vibrations e.g. from vehicles and accelerations due to accelerometer bouncing against other objects which produces mechanical resonance [12] by helping to trim the peaks but not edges 2.3 . The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighboring entries. The pattern of neighbors is called the ”window”, which slides, entry by entry, over the entire signal.

During data collection (in static mode) the accelerometer data was sampled at 100 Hz. Moreover, the application SensorStreamer can stream sensor data at 100 Hz, however the sampling rate of gyroscope and orientation data was 200 Hz. Hence, alternate values were selected for gyroscope and orientation data and hence subsampled to get all the values at the same frequency.

[2] Android Sensors Overview developer.android.com/guide/topics/sensors/sensors_overview.html

[3] iOS Sensors Overview developer.apple.com/documentation/uikit/uidevice

Eventually all the static data is combined into one single dataset to work on. And further it is split into two data sets namely **Train** and **Test** in the ratio of 75:25

2.2.2 Time Domain Feature Extraction

Earlier studies on fall detection mostly use simple thresholding of the sensory outputs (e.g., accelerations, rotational rates) because of its simplicity and low processing time. This approach is not sufficiently robust or reliable because there are different fall types and their nature shows variations for each individual. Furthermore, certain ADLs can be easily confused with falls. For improved robustness, we consider additional features of the recorded signals[17].

To do that we use the Windowing Approach. The rectangular window acts as a convenient way to take a part of the time series to operate on. To do that, we first look at the total acceleration acting on the device given by 2.3.

$$At = \sqrt{Ax^2 + Ay^2 + Az^2} \tag{2.3}$$

Where Ax, Ay, Az represents acceleration acting on x, y, z axis of the phone. The time index of this At or signal magnitude vector is determined for each activity record. This At shall act as a reference point to create the rectangular window. The width of this rectangular window is important and various studies have used widths varying from 1 to 7.5 seconds [10]. A window of size 4 seconds was chosen (also experimented with 2 seconds width window), which was found to be optimal in related work [17]. The window is centered around the maximum value of At; which shall allow to capture the impact start and also settling of the fall. At the same time index corresponding to maximum value of At, the window around gyroscope and orientation values across all three axis is also captured with the same window width. So in total we have window over 3 sensor values vix Accelerometer, Gyroscope, Orientation across 3 axis namely X,Y and Z making it a 9 column time series.

ADD FIGURE OF LONGER TIME SERIES AND THE WINDOW AND HENCE THE SHORTER TIME SERIES HERE.
To extract features from the windowed data, we look at time domain statistics such as mean, median, minimum value, maximum value, inter quartile range, skewness [13]. Both the Python and R have internal functions to calculate these parameters.

2.2.3 Frequency Domain Feature Extraction

Similar statistical features extracted of the time series in the frequency domain also assist in the model [13] [17]. To get the frequency domain values Discrete Fourier Transforms is used. If $x = (x_0, x_1, x_2, ..., x_{N-1})$ be a vector of samples of the signal, in this case accelerometer, gy-

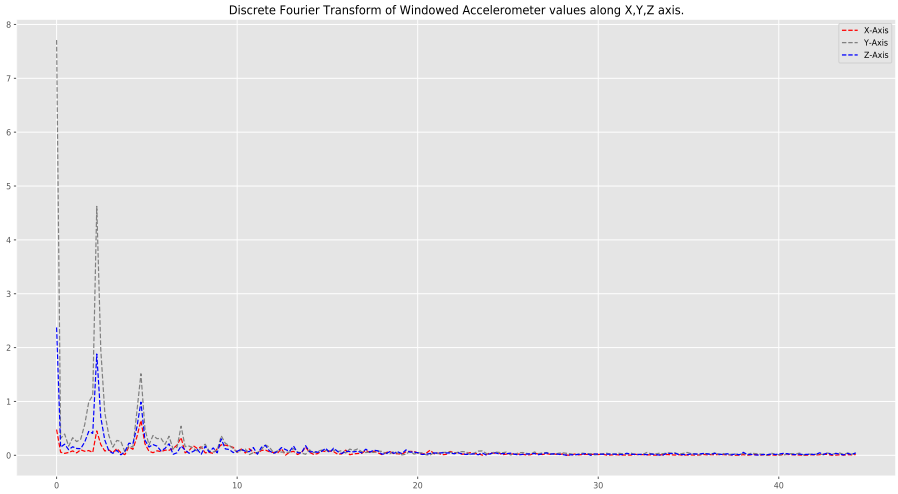

Figure 2.4: Discrete Fourier Transform of Windowed Accelerometer values along X,Y,Z axis. Only the positive frequency values are considered

roscope, orientation values. Its Discrete Fourier Transform (DFT) converts the signal from the time domain to the frequency domain by representing it as a combination of complex sinusoids with frequencies $f_k = k/N$ and corresponding coefficients

$$X_k = \sum_{n=0}^{N-1} x_n . \exp^{-i2\pi kn/N} \tag{2.4}$$

Where, $k = 0, ..., n1$. $|X_k|$ is the amplitude of a sinusoid with frequency f_k.

We take the statistical features such as mean, max, min, standard deviation, skewness, kurtosis of the sensor values across all the axis in the frequency domain. Figure 2.4 is a representation of accelerometer in frequency domain for action of jumps.

Hence, we create 7 features in time domain and 6 features in frequency domain, viz 13 features. For 3 sensor values across 3 axis and one extra for At of accelerometer we get in total 114 features. Moreover, age in years, gender, height in cms, weight in kilograms is also captured making the total number of features to 118. And as per 2.2 and 2.3 we get close to 2500 trials (whose only windowed values are captured) across 13 activites and 57 subjects.

With all the features in place, and 2500 observations of data, a dataframe is formed. The 'activity' with 13 unique activities can be classified using a classification algorithm.

To visualize the data, Principal Components analyses was performed. PCA was performed on the features extracted in the time domain domain 2.2.2 and frequency domain 2.2.3. PCA was

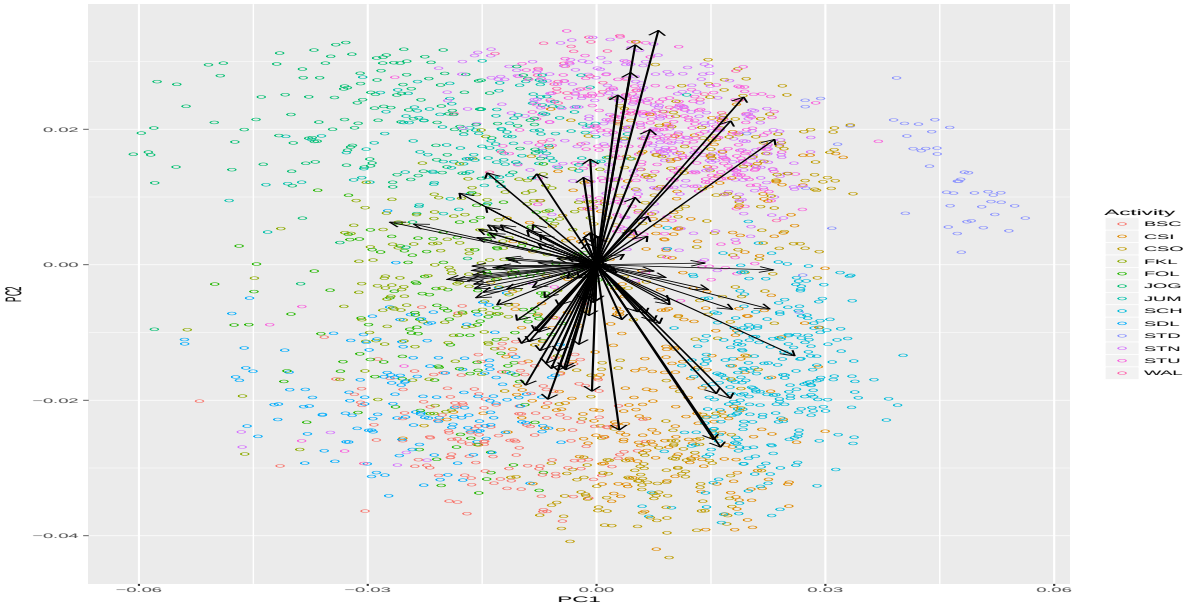

Figure 2.5: Principal Component Analysis. Figure shows first and second principal components on X and Y axis, and all 13 activities are color coded. The loadings are shown in Black.

used as a dimension reduction technique to explore the full available feature space in time and frequency domain features. PCA projects high-dimensional observations onto a lower dimensional subspace while maximally conserving variability between observations. The weight of each feature is commonly referred to as the loading for a particular component. The coordinates of the data points on each of the principal components are referred to as the scores. The principal components are ordered by the amount of variance that they explain in the original feature space. Figure 2.5
shows the projection of the observations of different activities recorded on first two principal components; all 13 activites are categorized. Where as in figure A.1 categories are grouped are grouped into Fall and ADLs. Tables 2.2 and 2.3 provide the activity meaning to the activity abbreviations used. As the classification is slightly clear with the first two principal components for the grouped *Fall* and *ADL* categories, however it is not very clear for all the 13 categories ungrouped; A.2 shows we need almost 20 PCs to explain 80% of the variance in the dataset.

2.3 Methods

2.3.1 Under Sampling

Before we started with classification, we observed that the classes are slightly imbalanced 2.6. In such situation, the predictive model developed using conventional machine learning algorithms could be biased and inaccurate. This is very common and degree of imbalance is very high in situations of fraud detection or anomaly detection. Different techniques are employed to deal with such situations [5]. One such way is to under sample the classes with very high

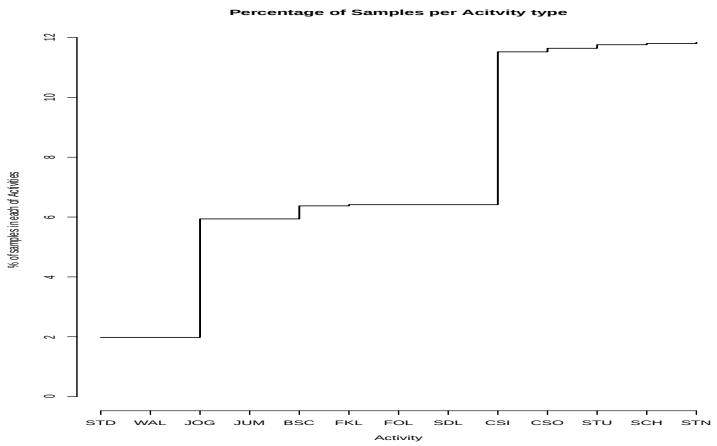

Figure 2.6: Percentage of Samples per Activity type. We observe, number of samples for CSO, STN,STU,SCH are almost 6 times that of STD, WAL, JOG

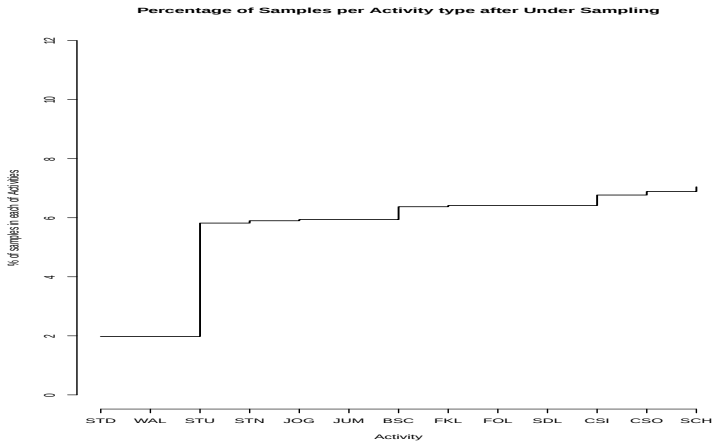

Figure 2.7: Percentage of Samples per Activity type After Under Sampling of CSO, STN,STU,SCH classes.

proportion of samples. It is also possible to over sample, the under represented classes, more detailed approaches are mentioned in [5].

After under sampling of classes of Activities namely CSO, STN,STU and SCH, we get representation in 2.7. However, there was no major change in accuracy of the final models over the test data. Hence, the undersampled approach was not followed.

The dataset was randomly split in the ratio of **75:25** for the **train** and **test** data sets. Further for the normalization of features *unity based normalization* was followed as given by 2.5

$$X' = \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{2.5}$$

Furthermore, The same parameters (X_{min},X_{max}) generated for the normalization of features for the *Train* dataset were used for the normalization of *Test* dataset, as in practice the test dataset

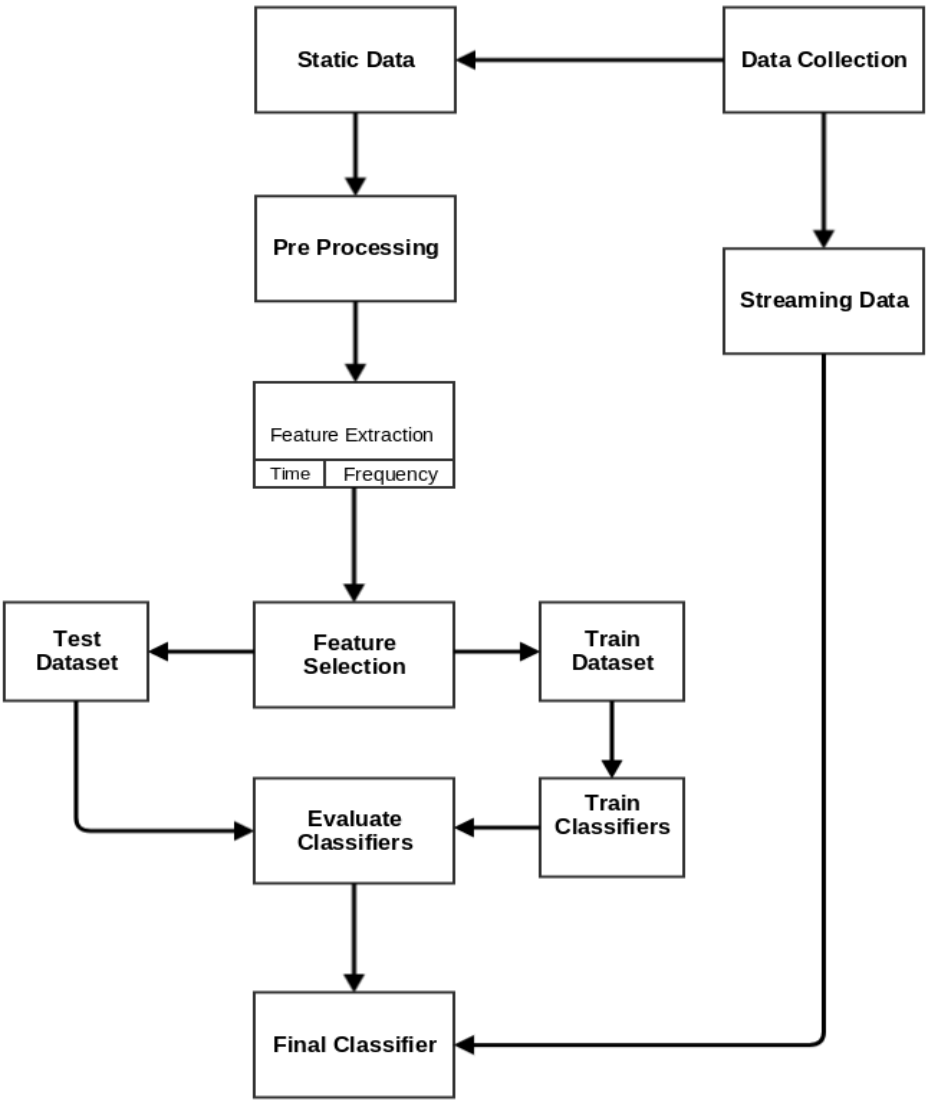

Figure 2.8: System Architecture

is not available. Moreover, the same parameters of normalization were used to normalize the streaming data for real time classification.

2.3.2 Classification models

For real time classification, the model was built on the static data. And this model was applied to the streaming data set. Few models were initially built on the static data set and evaluated to propose a final model which would be incorporated in the streaming setting, a schema is shown in 2.8.

The model was started with 117 predictors / classifiers. For the process of feature selection variable importance indicator in random forest is used. Furthermore, to select number of features to incorporate, a cross validation model with 10 folds is used. It shows the cross-validated

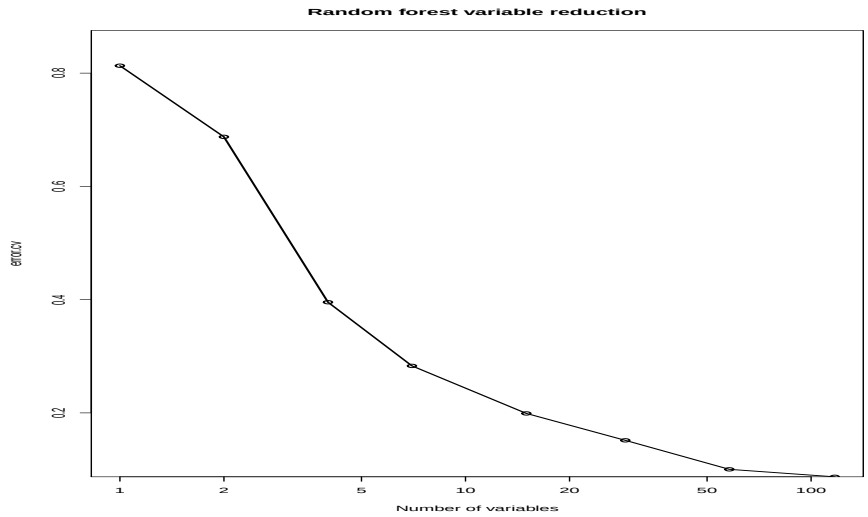

Figure 2.9: Random forest cross validated with 10 folds. With step of 2 for feature reduction based on Gini Index. Variable importance is (re-)assessed at each step of variable reduction

prediction performance of models with sequentially reduced number of predictors (ranked by variable importance) via a nested cross-validation procedure. It was set up with a step size of 2 with variable importance (re-)assessed at each step of variable reduction. From 2.9 it is said that 29 features to be selected, as it does not lead to major increase in CV error. Interestingly, only two features based on orientation were part of these 29 features namely skew of orientation in X axis of frequency domain and standard deviation of orientation in X axis of time domain. They were replaced by the next two features (non-orientation based) based on gyroscope values. This lead to major reduction in computations in the stream setting, as there was no further requirement to capture the orientation.

KNN

Given a positive integer K and a test observation x_0, the K nearest neighbor classifier first identifies the K points in the training data that are closest to x_0, represented by ν_0. It then estimates the conditional probability for class j as the fraction of points in ν_0 whose response values equal j [8]

$$Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \nu_0} I(y_i = j) \tag{2.6}$$

Finally, KNN applies Bayes rule and classifies the test observation x_0 to the class with the largest probability. KNN classifier was applied with $K = 1, 5, 20, 100$; which gave the accuracy on the test set $86.86\%, 82.6\%, 78.5\%, 68.2\%$ respectively. Where Accuracy is $(True\,positive + True\,Negative)\,/\,total\%$

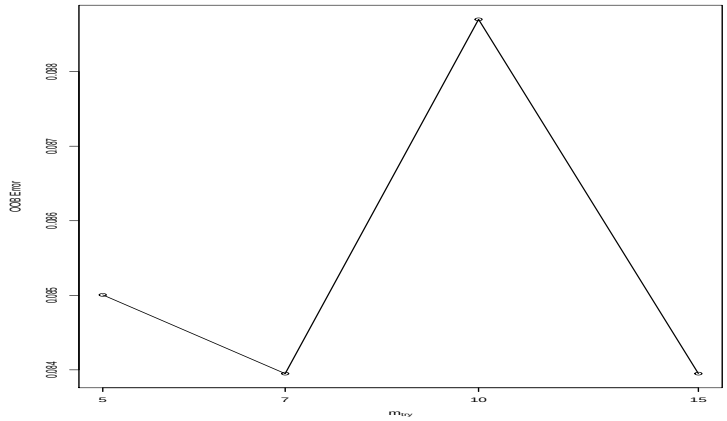

Figure 2.10: Tuning Random Forest

Random Forest

Random Forest when used for classification, obtains a class vote from each tree, and then classifies using majority vote. As recommendation :

- For classification, the default value for m is $\sqrt{p}$ and the minimum node size is one.
- For regression, the default value for m is $p/3$ and the minimum node size is five.

In [8] it is said "it is certainly true that increasing B [the number of trees] does not cause the random forest sequence to overfit", however in [6] it is also mentioned that "the average of fully grown trees can result in too rich a model, and incur unnecessary variance". Hence we vary the mtry value, i.e Number of variables randomly sampled as candidates at each split and observe the Out of Bag error. 2.10 clearly shows a varying error as $mtry$ is varied and is not minimum at $10 = \sqrt{117}$ that is square root of number of classifiers. The improvisation factor viz. the (relative) improvement in majority vote for the search to continue, was set to $1e^{-5}$ with 1000 tress used at the tuning step. The variable importance was based on the $GiniIndex$, defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{2.7}$$

Where, $\hat{p}_{mk}$ is proportion of training observations in the m^{th} region that are from the k^{th} class, and G is the measure of total variance across the K classes [8]. This lead to the classification accuracy of 90.00% on test data. However, the accuracy was decent, for activity 'Walk' out 11 observations 8 were misclassified in 'Stairs Up' and 'Stair Down' activity. Besides random forest, decision trees were also modeled whose accuracy on test dataset was 71.36%

Support Vector Machines

For classification with support vector machines - SVM, three types of $kernels$ were used namely $linear$, $radial$ and $polynomial$.

If the kernel is *linear*, then it is also known as Support Vector Classifier. In support vector classifier a hyperplane(s) is(are) created to differentiate the observations on p dimensions among the classes. Hyperplane is a sub-space of $p-1$ dimensions. It is chosen such that it separates *most* of the training observations into respective classes, but may misclassify a few observations. It is the solution to the optimization problem [8] :

$$maximize_{\beta_0,\beta_1....\beta_p,\epsilon_0,\epsilon_1,...,\epsilon_n} M \tag{2.8}$$

$$Subject\ to \sum_{j=1}^{p} \beta_j^2 = 1 \tag{2.9}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + ... + \beta_p x_{ip}) \geq M(1-\epsilon_i) \tag{2.10}$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C \tag{2.11}$$

Where, C is a tuning parameter (greater than 0), it bounds the sum of ϵ_i's and it determines the number and severity of the violations to the margin and to the hyperplane that needs to be tolerated, in other words it control the bias variance trade off. M is the width of the margin for classifier, ϵ_i are the $slackvariables$ that allow individual observations to be on the wrong side of the margin or the hyperplane. Once,2.8,2.9,2.10,2.11 are solved the test observation x^* can be classified depending on which side of hyperplane it lies. Another important property of these equations, is that only the observations that lie on the margin or that violate the margin will affect the hyperplane, they are know as *support vectors*[8].

We have 13 classes (on multi class classification with SVM is explained later), and 1894 observations in our train data dataset. To get the optimized value of C i.e cost, a cross-validation with 10 folds was performed. And $C = 0.1$ was set as the tuning parameter, and on building a model on that the **total accuracy achieved on Test data set was** 90.5%.

As noted in equations 2.8,2.9,2.10,2.11 it is a linear system. To address the non-linear behavior enlarging the feature space using quadratic, cubic or higher order polynomial functions can be done.

The support vector classifier can be extended by enlarging feature space using *kernels*. Such an extension is known as *Support Vector Machines*. The exact details of computation of SVM is out of scope of this thesis. However, solution to 2.8,2.9,2.10,2.11 support vector classifier problem involves only the inner product of the observations, it is defined as

$$\langle x_i, x_{i'} \rangle = \sum_{i=1}^{p} x_{ij} x_{i'j} \tag{2.12}$$

for two observations $x_i, x_{i'}$. The previously mentioned support vector classifier can also be

represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \tag{2.13}$$

where there are n parameters α_i, one per training observation. This implies, to estimate all the parameters, we need $\binom{n}{2}$ inner products between all pairs of training observations. Which is a huge number (in our train set case 1894*1893/2 = 1792671). However, it turns out that αi is a non-zero real number only for the *support vectors* in the solution, viz for all the others it is zero. So if S is the collection of indices of these support points, equation 2.13 can be re-written as

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle \tag{2.14}$$

Which drastically reduces the computation, as $i \in S$ is a far smaller number compares to n. This inner product used can be replaced by a *generalization* [8] of the inner product of the form

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{2.15}$$

where K is a generic function which can be referred as *kernel*. For the kernel format mentioned in 2.15, refers to linear kernel as the classifier is linear in the features; it essentially quantifies the similarity of a pair of observations using Pearson (standard) correlation. For the linear kernel fit for classification of 13 classes with n=1894, there were **731 support vectors**.

For the Radial Kernel SVM, kernel takes the form of

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{2.16}$$

Where, γ is a positive constant. If the given test observation $x^* = (x_1^*, x_2^*, x_3^*, x_4^*, ..., x_p^*)^T$ is far from a training observation x_i in terms of Euclidean distance, then the summation in 2.16 will be large and hence the LHS of equation will be small as the summation is exp to the power of negative number. Which means in 2.17, role of x_i will be insignificant.

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i) \tag{2.17}$$

which is a generic representation of support vector machine.

To find the value of γ in the models, a Grid Search was used along with various values for Cost. Over this grid a 10 fold cross validation was fit and for each combination of γ and $Cost$ value a cross validation error was established. It was found $Cost = 10$ and $\gamma = 0.01$ for Radial kernel classification. Number of support vectors were 1348 for a cumulative of 13 classes. This model

provided a total accuracy of 93.67% on the test data set.

For the polynomial kernel, kernel takes the form of

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \tag{2.18}$$

Where d is a positive integer, and this is known as a polynomial kernel of degree d. In case of polynomial kernel too a grid search was used to find the optimum value of d and the $Cost$ using a 10 fold cross validation. It was found $Cost = 2$ and $d = 3$ optimum for polynomial kernel classification. And model resulted with a total accuracy of 89.7% on the test data set with 1420 number of support vectors for all the classes put together.
For multiclass-classification with k levels, k¿2, libsvm uses the one-against-one-approach, in which k(k-1)/2 binary classifiers are trained; the appropriate class is found by a voting scheme, in our scenario with k=13, 78 binary classifiers would be trained.

2.4 Model Evaluation

For model evaluation procedure, the situation is of multi class classification models. The classification model can produce the following four possible outcomes [3]

1. True Positive (TP) when a system properly detects a fall when fall has occurred.
2. False Positive (FP) when a system detects a fall when no fall has occurred.
3. True Negative (TN) when a system detects no fall when no fall has occurred.
4. False Negative (FN) when a system detects no fall when a fall has occurred.

Confusion matrix for each of the model was created, for the test data set A.5 A.1. The confusion matrix provides a tabular summary of the actual class labels vs. the predicted ones. To calculate the over all **accuracy** of the model, fraction of instances that are correctly classified was calculated.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\sum diagonal\ elements(Confusion\ Matrix)}{Total\ num\ of\ elements\ in\ Confusion\ Matrix} \tag{2.19}$$

In order to assess the performance with respect to every class in the dataset, we will compute common per-class metrics such as precision and recall (also F-1 score). These metrics are particularly useful when the class labels are not uniformly distributed.2.6 shows the imbalanced nature of the dataset, for instance, number of samples for activities such as Walk, Stand, Jog are very low in comparison to Stairs Up, Stairs Down. Hence, the over all accuracy might be

misleading in such a scenario, as it could predict the dominant class most of the time and still achieve a relatively high overall accuracy but very low precision or recall for other classes.

Precision is defined as the fraction of correct predictions for a certain class,

$$Precision = \frac{TP}{TP + FP} \tag{2.20}$$

whereas **recall** is the fraction of instances of a class that were correctly predicted.

$$Recall = Sensitivity = \frac{TP}{TP + FN} \tag{2.21}$$

Notice that there is an obvious trade off between these 2 metrics. When a classifier attempts to predict one class, say class Stairs Up, most of the time, it will achieve a high recall for Stairs Up (most of the instances of that class will be identified). However, instances of other classes will most likely be incorrectly predicted as Stairs Up in that process, resulting in a lower precision for Stairs Up. The per-class metrics can be averaged over all the classes resulting in macro-averaged precision and recall.

The 2.4 summarizes the precision values for all the classes, and it seems the support vector model with radial kernel outperforms the other. Also in 2.5 table summarizes the recall values for all the classes and SVM with radial kernel is a better fit. The KNN models performed poorly with 0.86, 0.82, 0.78 over all accuracies for k=1,5,20.

	PRECISION			
	Random Forest	SVM Linear	SVM Radial	SVM Polynomial
BSC	0.8684211	0.9230769	0.8571429	0.9230769
CSI	0.8507463	0.8852459	0.984127	0.7971014
CSO	0.8554217	0.8554217	0.9047619	0.875
FKL	0.8913043	0.8125	0.9545455	0.8695652
FOL	0.8378378	0.8	0.9090909	0.8823529
JOG	0.9756098	0.974359	1	1
JUM	1	1	0.9117647	1
SCH	0.974026	1	1	0.987013
SDL	0.8780488	0.9268293	0.8695652	0.9736842
STD	1	1	1	1
STN	0.9390244	0.8823529	0.9156627	0.8275862
STU	0.9166667	0.8666667	0.9444444	0.8421053
WAL	0.8333333	0.8333333	1	1
			Macro Avg Precision	
	0.9092646308	0.9045989	**0.9423927077**	0.9213450077
			Micro Avg Precision	
	0.90822785	0.90031646	**0.93670886**	0.89556962
KAPPA	0.89859186	0.88983515	0.93005755	0.88449845

Table 2.4: Model evaluation parameters. It is Observed SVM with Radial kernel has a better macro and micro average precision

	RECALL			
	Random Forest	SVM Linear	SVM Radial	SVM Polynomial
BSC	0.825	0.9	0.9	0.9
CSI	0.8769231	0.8307692	0.9538462	0.8461538
CSO	0.9102564	0.9102564	0.974359	0.8974359
FKL	0.9318182	0.8863636	0.9545455	0.9090909
FOL	0.9117647	0.8235294	0.8823529	0.8823529
JOG	0.9756098	0.9268293	0.9268293	1
JUM	1	1	1	1
SCH	0.9615385	0.9871795	0.9871795	0.974359
SDL	0.75	0.7916667	0.8333333	0.7708333
STD	0.8461538	0.9230769	0.8461538	0.9230769
STN	1	0.974026	0.987013	0.9350649
STU	0.9166667	0.9027778	0.9444444	0.8888889
WAL	0.4545455	0.4545455	0.4545455	0.1818182
			Macro Avg Recall	
	0.8738674385	0.8700784846	**0.8957386462**	0.8545442077
			Micro Avg Recall	
	0.90822785	0.90031646	**0.93670886**	0.89556962

Table 2.5: Model evaluation parameters. It is Observed SVM with Radial kernel has a better macro and micro average average recall

3

Apache Spark

3.1 Spark Overview

At a high level, every Spark application consists of a driver program that runs the users main function and executes various parallel operations on a cluster. The main abstraction Spark provides is a resilient distributed dataset (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. RDDs are created by starting with a file in the Hadoop file system (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it. Users may also ask Spark to persist an RDD in memory, allowing it to be reused efficiently across parallel operations. Finally, RDDs automatically recover from node failures [16] [14].

A second abstraction in Spark is shared variables that can be used in parallel operations. By default, when Spark runs a function in parallel as a set of tasks on different nodes, it ships a copy of each variable used in the function to each task. Sometimes, a variable needs to be shared across tasks, or between tasks and the driver program. Spark supports two types of shared variables: broadcast variables, which can be used to cache a value in memory on all nodes, and accumulators, which are variables that are only added to, such as counters and sums [16] [14].

Figure 3.1: Apache Spark Streaming.

3.2 Spark Streaming

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like Kafka, Flume, Kinesis, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions like map, reduce, join and window. Finally, processed data can be pushed out to filesystems, databases, and live dashboards [15].

The main component of spark streaming is D-Streams. D-Streams avoid the problems with traditional stream processing by structuring computations as a set of short, stateless, deterministic tasks instead of continuous, stateful operators. They then store the state in memory across tasks as fault-tolerant data structures (RDDs) that can be recomputed deterministically. Decomposing computations into short tasks exposes dependencies at a fine granularity and allows powerful recovery techniques like parallel recovery and speculation. Beyond fault tolerance, the D-Stream model gives other benefits, such as powerful unification with batch processing [14].

As part of this thesis, spark streaming is run in a **stand alone mode**, on a system with 4 cores, with processor AMD A4-6210 APU with AMD Radeon R3 Graphics and 6.8 Gb memory running ubuntu 16.04 OS. The stream of sensor data from Android Phone was streamed to spark via TCP protocol. As mentioned in 2.3.2 accelerometer and gyroscope data were needed for the classifier as input. The batch interval for the D-Stream was set to 1 second. Spark Streaming API was run in Python 3.5.2 and also the final classifier was implemented in python.

The stream data was sent in JSON objects which was further parsed into a usable structure.Spark Streaming provides windowed computations, which allow to apply transformations over a sliding window of data 3.2. As the window size for the classification was set at 4 seconds similar sized windowing is applied in the streaming context. Though the output of the stream can be sent to a dashboard, no dashboard was created in this work.

For implementing the classification, windowed accelerometer and gyroscope values were transformed *foreachRDD* of data to give the Time domain and Frequency domain features as mentioned earlier. These features were normalized using unity based normalization method 2.5

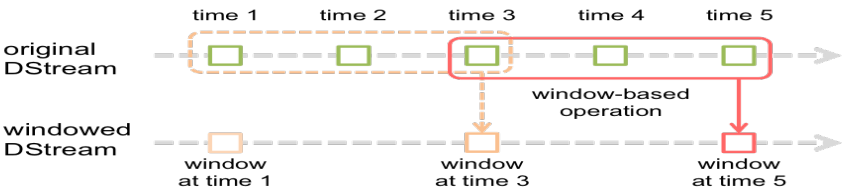

Figure 3.2: Illustration : every time the window slides over a source DStream, the source RDDs that fall within the window are combined and operated upon to produce the RDDs of the windowed DStream.

using the same paramaters of normalization as that was used for *training data*.

As evaluated in 2.4 model evaluation section, according to 2.4 Support vector Machine with Radial Kernel was the most optimum. Hence, the same was implemented with $\gamma = 0.01$ and $Cost = 10$ in Python using SciPy [1]. For simplicity and also as the number of observations captured in 4 second window at 100 Hz, would be max of 400, the model application for prediction of class was carried out in the *master*. No master-slave cluster was configured, however, all the cores of the CPU were utilized. The output of the classification is obtained at almost per second interval on the console.

4

Discussion

In this thesis we proposed a method to classify the ADL activities and Fall activities of a person in *close real time*. The classification model chosen was radial kernel support vector machine. In the works of [10, Pranesh Vallabh and Reza Malekian and Ning Ye and Dijana Capeska Bogatinoska] the classification was carried out between the ADLs and FALL with two class classification in the ”MobiFall” dataset and a filter rank based system was implemented to extract the top five features in order to optimize algorithms dimensionality.And the k-NN with k equal to 5 achieved an 87.5% accuracy which resulted in the highest accuracy when compared to the other classification methods. However in the work of [11, George Vavoulas and Charikleia Chatzaki and Thodoris Malliotakis and Matthew Pediaditis and Manolis Tsiknakis] nine different types of Activities of Daily Living (ADLs) were classified (No falls were considered). The best overall accuracy of 99.88% was achieved when using the IBk classification algorithm on the MobiAct dataset. In both the works, features were extracted only from accelerometer readings.

In this thesis an attempt was made to classify the four types of falls and nine types of ADLs totalling to 13 class classification 2.2 2.3 using the ”MobiAct” dataset. The best overall accuracy of 93.68% was achieved using radial kernel SVM classifier. Moreover, not only accelerometer readings but also gyroscope values were considered for feature extraction in this study.

Despite the best efforts, as the Falls still remain simulated over a population of 57 volunteers in the age group of 20 to 47 years, extending the results for relatively elderly people might not

suffice. Hence, a more detailed data collection involving elderly people and capturing natural falls may be taken up. In future work, a better evaluation of models in the streaming data format may be carried out. And for spark streaming, moving from TCP connection to a more reliable connection can be thought of. For the visualization on the streaming data and classification a real time dashboard may be designed with an *action* sequence such as notifying someone in case of a fall detection. For model up-gradation online learning can be explored to make use of streaming data to train the model and also to make custom models per subject.

References

[1] SciPy: Open source scientific tools for Python, 2014. [Online; accessed ¡today¿].

[2] F. Abyarjoo. Sensor fusion for effective hand motion detection. 2015.

[3] B. Aguiar, T. Rocha, J. Silva, and I. Sousa. Accelerometer-based fall detection for smartphones. In *Medical Measurements and Applications (MeMeA), 2014 IEEE International Symposium on*, pages 1–6. IEEE, 2014.

[4] Y. Cao, Y. Yang, and W. Liu. E-falld: A fall detection system using android-based smartphone. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 1509–1513. IEEE, 2012.

[5] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[6] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[7] Y. He, Y. Li, and S.-D. Bao. Fall detection by built-in tri-accelerometer of smartphone. In *Biomedical and Health Informatics (BHI), 2012 IEEE-EMBS International Conference on*, pages 184–187. IEEE, 2012.

[8] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[9] D. K. Shaeffer. Mems inertial sensors: A tutorial overview. *IEEE Communications Magazine*, 51(4):100–109, 2013.

[10] P. Vallabh, R. Malekian, N. Ye, and D. C. Bogatinoska. Fall detection using machine learning algorithms. *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–9, 2016.

[11] G. Vavoulas, C. Chatzaki, T. Malliotakis, M. Pediaditis, and M. Tsiknakis. The mobiact dataset: Recognition of activities of daily living using smartphones. In *Proceedings of the International Conference on Information and Communication Technologies for Ageing*

Well and e-Health - Volume 1: ICT4AWE, (ICT4AGEINGWELL 2016), pages 143–151. INSTICC, ScitePress, 2016.

[12] G. Vavoulas, M. Pediaditis, E. G. Spanakis, and M. Tsiknakis. The mobifall dataset: An initial evaluation of fall detection algorithms using smartphones. In *13th IEEE International Conference on BioInformatics and BioEngineering*, pages 1–4, Nov 2013.

[13] J. Wannenburg and R. Malekian. Physical activity recognition from smartphone accelerometer data for user context awareness sensing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP(99):1–8, 2017.

[14] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 423–438, New York, NY, USA, 2013. ACM.

[15] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica. Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. In *Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Ccomputing*, HotCloud'12, pages 10–10, Berkeley, CA, USA, 2012. USENIX Association.

[16] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, Oct. 2016.

[17] A. T. zdemir and B. Barshan. Detecting falls with wearable sensors using machine learning techniques. *Sensors*, 14(6):10691–10708, 2014.

Appendices

A

Appendix 1

The freely available raw data has been placed in cloud for any one to access [1].
Confusion Matrices of test data set illustrating predicted class vs actual class for different models.

[1] Raw Data on Cloud https://drive.google.com/drive/u/1/folders/0B8QGwf0_SNEFcHZLNVRla2pWUWM

	Test Data (Actual Data)												
Pred	BSC	CSI	CSO	FKL	FOL	JOG	JUM	SCH	SDL	STD	STN	STU	WAL
BSC	31	0	0	0	1	0	0	0	4	0	0	0	0
CSI	2	57	1	0	0	0	0	2	2	1	1	1	0
CSO	0	1	75	0	0	0	0	0	0	0	0	0	1
FKL	1	0	0	38	7	0	0	0	1	0	0	0	0
FOL	2	0	0	5	23	0	0	0	5	0	0	0	0
JOG	0	0	0	0	0	37	1	0	0	0	0	0	1
JUM	0	0	0	0	0	2	29	0	0	0	0	0	0
SCH	1	1	0	0	0	0	0	76	0	0	0	0	0
SDL	3	0	0	1	3	0	0	0	36	0	0	0	0
STD	0	0	0	0	0	0	0	0	0	10	0	0	0
STN	0	2	2	0	0	1	1	0	0	1	72	6	3
STU	0	4	0	0	0	0	0	0	0	1	2	63	4
WAL	0	0	0	0	0	1	0	0	0	0	2	2	2

Table A.1: KNN classifier, Test Data Confusion Matrix

	Test Data (Actual Data)												
Pred	BSC	CSI	CSO	FKL	FOL	JOG	JUM	SCH	SDL	STD	STN	STU	WAL
BSC	27	0	0	0	0	0	0	0	4	0	0	0	0
CSI	4	38	7	0	0	0	0	9	2	0	1	3	0
CSO	0	15	63	0	0	0	0	4	1	1	0	9	0
FKL	1	0	0	18	4	0	1	0	6	0	0	0	0
FOL	1	0	0	20	29	0	0	0	3	0	0	0	0
JOG	0	0	0	1	0	32	3	0	0	0	3	1	0
JUM	0	0	0	0	0	4	27	0	0	0	0	0	0
SCH	0	4	1	0	0	0	0	65	0	1	0	0	0
SDL	7	0	0	0	1	3	0	0	32	0	0	0	0
STD	0	0	0	0	0	0	0	0	0	11	0	1	0
STN	0	7	1	3	0	0	0	0	0	0	69	18	4
STU	0	1	6	2	0	2	0	0	0	0	4	40	7
WAL	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.2: Decision Tree Classifier, Test Data Confusion Matrix

	Test Data (Actual Data)												
Pred	BSC	CSI	CSO	FKL	FOL	JOG	JUM	SCH	SDL	STD	STN	STU	WAL
BSC	33	0	0	0	0	0	0	0	5	0	0	0	0
CSI	2	57	5	0	0	0	0	2	1	0	0	0	0
CSO	0	7	71	0	0	0	0	1	1	0	0	3	0
FKL	1	0	0	41	2	0	0	0	2	0	0	0	0
FOL	0	0	0	3	31	0	0	0	3	0	0	0	0
JOG	0	0	0	0	0	40	0	0	0	0	0	1	0
JUM	0	0	0	0	0	0	31	0	0	0	0	0	0
SCH	0	1	0	0	0	0	0	75	0	1	0	0	0
SDL	4	0	0	0	1	0	0	0	36	0	0	0	0
STD	0	0	0	0	0	0	0	0	0	11	0	0	0
STN	0	0	0	0	0	0	0	0	0	0	77	2	3
STU	0	0	2	0	0	0	0	0	0	1	0	66	3
WAL	0	0	0	0	0	1	0	0	0	0	0	0	5

Table A.3: Random Forest classifier, Test Data Confusion Matrix

	Test Data (Actual Data)												
Pred	**BSC**	CSI	CSO	FKL	FOL	JOG	JUM	SCH	SDL	STD	STN	STU	WAL
BSC	36	0	0	0	0	0	0	0	3	0	0	0	0
CSI	1	54	4	0	0	0	0	1	1	0	0	0	0
CSO	0	10	71	0	0	0	0	0	1	0	0	1	0
FKL	1	0	0	39	5	0	0	0	3	0	0	0	0
FOL	0	0	0	5	28	0	0	0	2	0	0	0	0
JOG	0	0	0	0	0	38	0	0	0	0	0	0	1
JUM	0	0	0	0	0	0	31	0	0	0	0	0	0
SCH	0	0	0	0	0	0	0	77	0	0	0	0	0
SDL	2	0	0	0	1	0	0	0	38	0	0	0	0
STD	0	0	0	0	0	0	0	0	0	12	0	0	0
STN	0	0	0	0	0	2	0	0	0	0	75	6	2
STU	0	1	3	0	0	1	0	0	0	1	1	65	3
WAL	0	0	0	0	0	0	0	0	0	0	1	0	5

Table A.4: Support Vector Machine with Linear Kernel classifier, Test Data Confusion Matrix

	Test Data (Actual Data)												
Pred	**BSC**	CSI	CSO	FKL	FOL	JOG	JUM	SCH	SDL	STD	STN	STU	WAL
BSC	36	0	0	0	0	0	0	0	6	0	0	0	0
CSI	0	62	0	0	0	0	0	0	0	1	0	0	0
CSO	1	2	76	0	0	0	0	1	0	1	1	1	1
FKL	0	0	0	42	2	0	0	0	0	0	0	0	0
FOL	0	0	0	1	30	0	0	0	2	0	0	0	0
JOG	0	0	0	0	0	38	0	0	0	0	0	0	0
JUM	0	0	0	0	0	3	31	0	0	0	0	0	0
SCH	0	0	0	0	0	0	0	77	0	0	0	0	0
SDL	3	0	0	1	2	0	0	0	40	0	0	0	0
STD	0	0	0	0	0	0	0	0	0	11	0	0	0
STN	0	1	1	0	0	0	0	0	0	0	76	3	2
STU	0	0	1	0	0	0	0	0	0	0	0	68	3
WAL	0	0	0	0	0	0	0	0	0	0	0	0	5

*Table A.5: Support Vector Machine with **Radial Kernel classifier**, Test Data Confusion Matrix*

	Test Data (Actual Data)												
Pred	**BSC**	CSI	CSO	FKL	FOL	JOG	JUM	SCH	SDL	STD	STN	STU	WAL
BSC	36	0	0	0	0	0	0	0	3	0	0	0	0
CSI	2	55	4	1	0	0	0	2	4	0	1	0	0
CSO	0	7	70	0	0	0	0	0	0	0	0	3	0
FKL	0	0	0	40	4	0	0	0	2	0	0	0	0
FOL	0	0	0	3	30	0	0	0	1	0	0	0	0
JOG	0	0	0	0	0	41	0	0	0	0	0	0	0
JUM	0	0	0	0	0	0	31	0	0	0	0	0	0
SCH	0	0	0	0	0	0	0	76	0	0	1	0	0
SDL	1	0	0	0	0	0	0	0	37	0	0	0	0
STD	0	0	0	0	0	0	0	0	0	12	0	0	0
STN	1	3	2	0	0	0	0	0	1	0	72	5	3
STU	0	0	2	0	0	0	0	0	0	1	3	64	6
WAL	0	0	0	0	0	0	0	0	0	0	0	0	2

Table A.6: Support Vector Machine with Polynomial Kernel classifier, Test Data Confusion Matrix

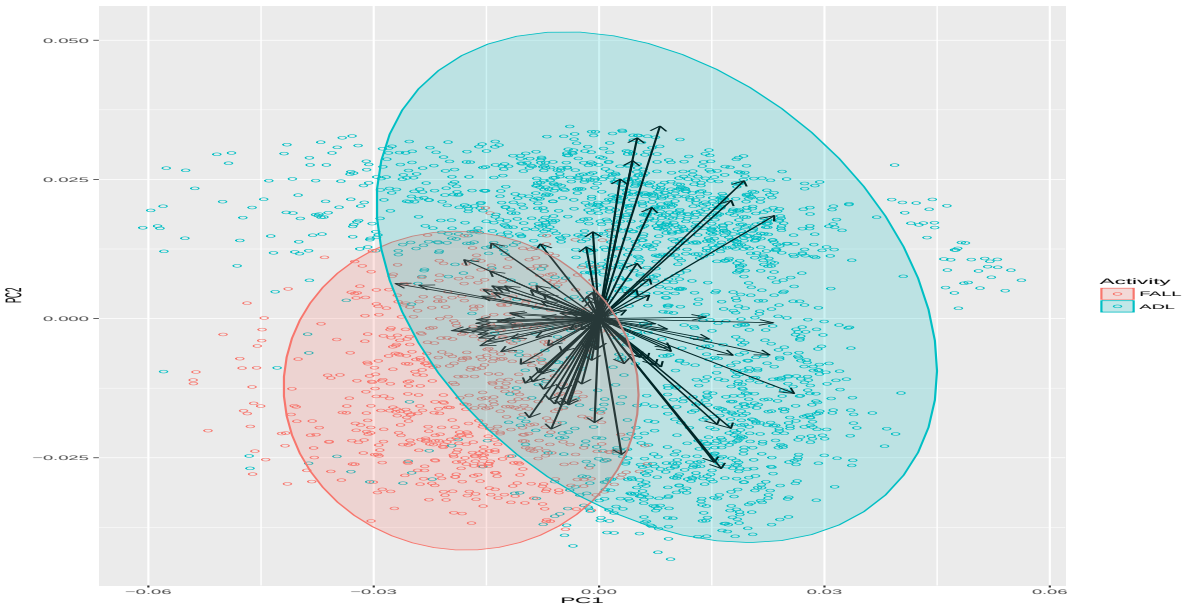

Figure A.1: Principal Component Analysis. Figure shows first and second principal components on X and Y axis, and activities are color coded. The activities are merged into two categories of Fall and ADLs.

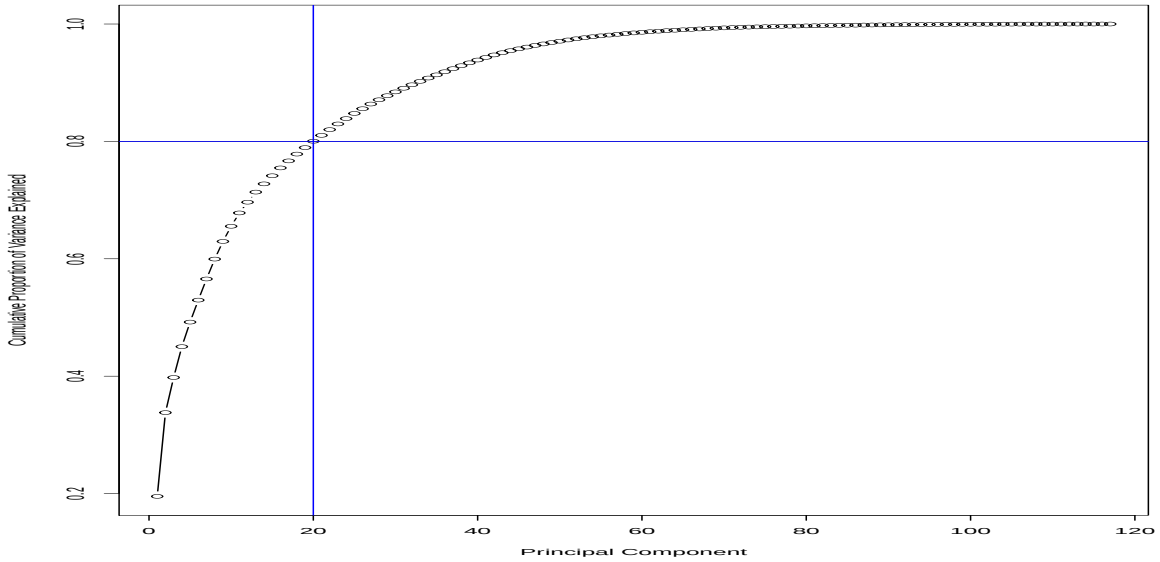

Figure A.2: Principal Component Analysis. Cumulative Proportion Of Variances.

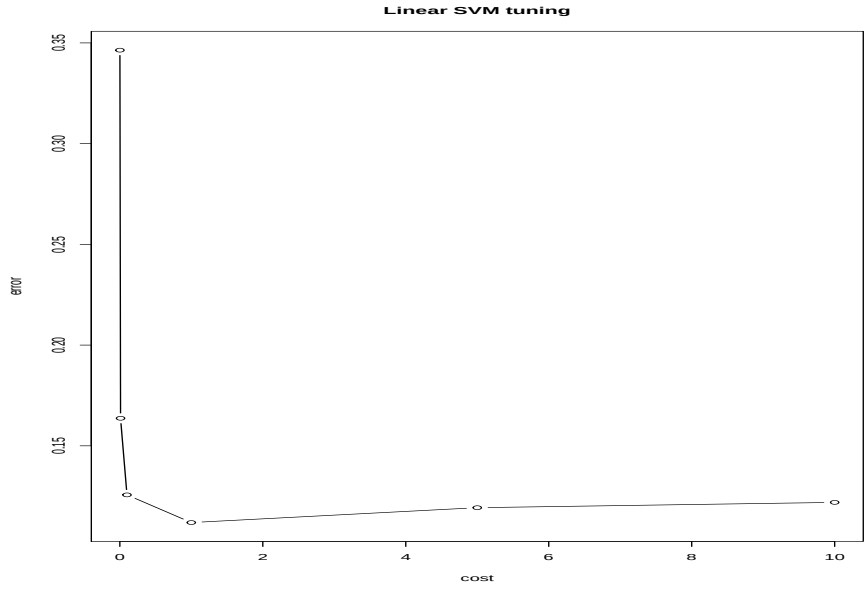

Figure A.3: Support Vector Machine with Linear Kernel. The plot shows how the 'Cost' parameter can be tuned to get minimum error with 10 fold cross validation.

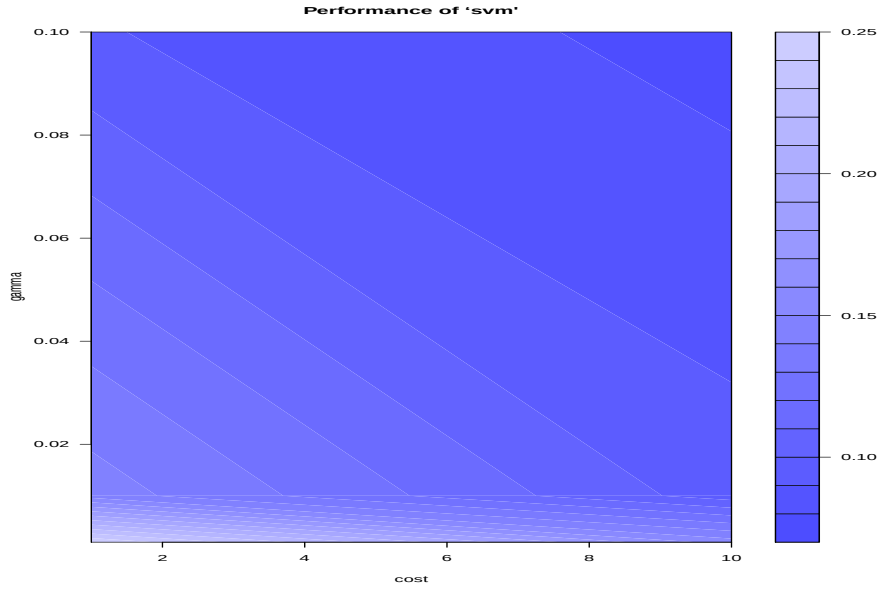

Figure A.4: Support Vector Machine with Radial Kernel. The plot shows how the 'Cost' and γ parameter can be tuned to get minimum error with 10 fold cross validation. A cost of 10 and $\gamma = 0.1$ is set for minimum error.

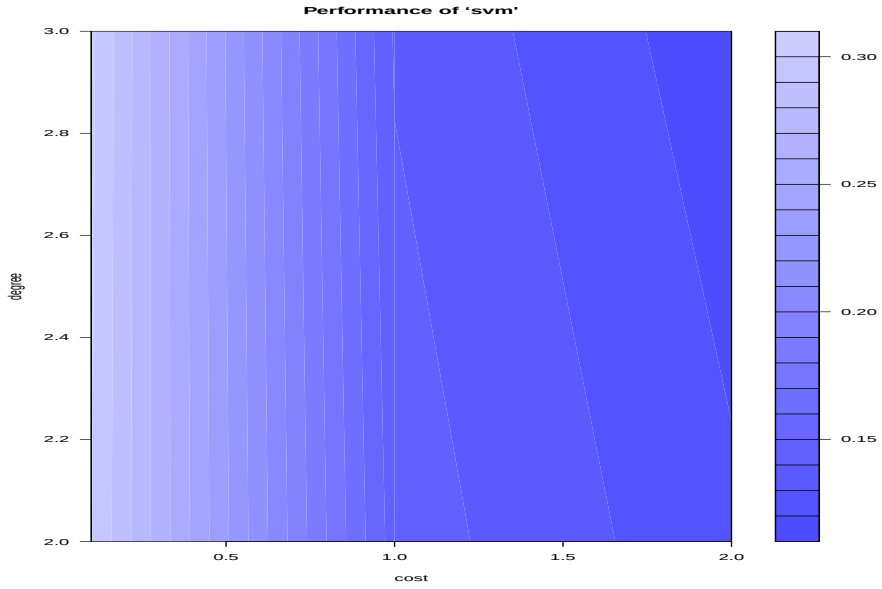

Figure A.5: Support Vector Machine with Radial Kernel. The plot shows how the 'Cost' and $Polynomial_Degree$ parameter can be tuned to get minimum error with 10 fold cross validation. A cost of 2 and $Degree = 3$ is set for minimum error.