UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING & INFORMATICS

IMPROVING ACCURACY IN ARRIVAL TIME PREDICTION USING SUPPORT VECTOR REGRESSION

(A CASE STUDY OF TRUCKS PASSING ALONG UHURU HIGHWAY NAIROBI)

BY

BRIAN OTIENO ONYANGO

REG NO: P58/63831/2011

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

SUPERVISOR: DR. LAWRENCE MUCHEMI

JUNE 2015

DECLARATION

This research project, as presented in this report is my original work and has not been presented in any other university award.

Signature: ………………………………… Date: ………/………/…………

Brian Otieno Onyango

Reg. No: P58/63831/2011

This research project has been submitted in partial fulfilment of the requirements for the Degree of Master of Science in Computer Science at the University of Nairobi with my approval as the university supervisor.

Signature: ………………………………… Date: ………/………/…………

Dr. Eng. Lawrence Muchemi

University Supervisor

ABSTRACT

Travel time prediction is vital to the development of reliable travel information systems. In this research project, Support vector regression (SVR) algorithms is applied to develop travel-time prediction model and show that it can lower the prediction error with the use of correct parameter choice and improve the accuracy of the prediction model as measured through the use of RMSE and MAPE. The purpose of this research project was to create a model that will predict the vehicle arrival time between two clearly defined sets of road locations. It was demonstrated that the feasibility of applying SVR algorithm in developing a travel-time prediction model performs well for travel time predictions. A simple web based system prototype was built to demonstrate how the results can be used to implement a working system for everyday traffic consumption.

A case study was designed in which over 4000 datasets across five sections was collected during the study period along Uhuru Highway, Nairobi, Kenya. Taking into account the controlled movements and stops of trucks, the mechanism of travel time prediction proposed in this study uses data mobility-coordinates with timestamp to learn then estimate the time a truck will arrive at a given predefined location given the time and place of departure.

The developed time predictions model demonstrates much success in estimating truck arrival time for the sections under study and performs much better on average compared to linear regression across the five sections tested. The model is able to predict arrival time to within 2 minutes accuracy and averaged 18.01% in RMSE error across the five sections of study. On average a MAPE of less than 1% across the five sections was achieved as well. Zegeye and Null researched on the use of dynamic artificial neural network travel time prediction model for buses using only GPS data in Brazil and got an average MAPE result of 18.3% (Zegeye & Nall, 2014). In 2003, Wei and co researched on the use of SVR for travel time prediction for long distance travel along a freeway and averaged 13.91% in RMSE (Wei, Wu, Ming-Hua , Chang, & Ho, 2003).
The exploration of three different predictive algorithms i.e. RBF, linear SVR and linear regression provide the research with alternative models that can be adopted for each sections. On average the RBF kernel algorithm (SVR) preformed much better compared to the other two models across the five sections. The web based applications used as prototype to demonstrate the applicability of this project makes a strong case that this project can be presented as a commercial venture.

ACKNOWLEDGEMENT

I thank God almighty who has been with me throughout the research study and the entire program.

I would like to express my gratitude to my Supervisor, Dr. Lawrence Muchemi, for providing me with an opportunity to work under his guidance. Without his great knowledge depth, availability, recommendations and attention to details, this project would not have been materialized.

I would also like to express my sincere appreciation to my wife who provided guidance with regard to financial management for this project. Her support towards completing this project was very uplifting.

The project would not have been complete without the support of my friends, work supervisor and data collection team, who with their constant support and valuable feedback helped me to continue with the good work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SVM	Support Vector Machine
SVR	Support Vector Regression
ETA	Estimated Time of Arrival
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
RBF	Radial Basis Function
ANN	Artificial Neural Networks
GPS	Global Positioning System
MAPE	Mean Absolute Percentage Error

DEFINITION OF KEY TERMS

Support Vector Machine: Support vector machines (SVMs) are a set of related supervised learning methods that analyze data and recognize patterns, used for classification (machine learning) classification and regression analysis

Support Vector Regression (SVR): A support vector machine method used to solve regression problems. Support Vector Regression depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction.

Estimated Time of Arrival: The estimated time of arrival or ETA (also ETOA) is a measure of when a ship, vehicle, aircraft, cargo, emergency service or computer file is expected to arrive at a certain place.

Traffic congestion map: A traffic congestion map is a graphical, real-time or near-real-time representation of traffic flow for some particular area. Data is typically collected via anonymous GPS data points and loop sensors embedded in the roadways, then processed by computer at a central facility and distributed as a map view to users.

Web Application: A web application or web app is any application software that runs in a web browser and is created in a browser-supported programming language (such as the combination of PHP, JavaScript, HTML and CSS) and relies on a common web browser to render the application.

Arrival Time: The time a truck arrives at a specific defined section.

Artificial Neural Networks (ANNs): Statistical learning models motivated by biological neural networks and are used to approximate functions that depend generally on unknown number of inputs.

CHAPTER I

INTRODUCTION

1.1 Background

Accelerating traffic growth over the past decades has been a threat to the quality of life of people in many countries. Traffic congestion leads to decrease in accessibility, travel time loss and air pollution. In developed countries most people still use private vehicles while in developing countries the level of vehicle ownership is rising at a faster rate.

Taking a metropolitan city like Nairobi as an example, the government approximations reveals that traffic jams cost 50 million shillings ($578,000) a day in lost productivity in the city. International Business Machines Corp.'s Commuter Pain survey (Welsh, 2011) reveals that the city's roads were the world's fourth-most congested. Unpredictable and growing traffic jam is a drawback to the quality of life of people in many urban centers, including the city of Nairobi, over the past few decades.

Even though many town areas in some parts of developed world are providing real-time travel time information on freeways, there are still challenges in providing accurate real-time travel time information on managed urban streets because of the nature of urban traffic. This is even worse in developing countries considering the undisciplined traffic and lack of available data that can be used to improve technology for managing traffic congestion. Therefore, to provide passengers with real-time travel time information, there is need to develop a good predictive algorithm model that can predict travel time with reasonable accuracy.

1.2 Accelerating Growth

Kenya is one of the most rapidly urbanizing places in the world. Compared to Europe that had 100 years to adjust to the number of vehicles and urbanization that is happening in Africa over 10 or 20 years, this is taking people by surprise. The growth is much faster than how it can be responded to. Nairobi City has road capacity for a population one-third its current size of 3.1 million -- a figure forecast to balloon to 40 million by 2050, the equivalent of the entire country's current population. The infrastructure and traffic management has not matched the more than doubling of the number of vehicles on Nairobi's roads since 2012 to 700,000, and moreover the city must prepare for as many as 9 million car users by 2050.

The public-transit system in Nairobi is part of the problem and traffic enforcement doesn't offer much relief. In the Kenyan capital, traffic crawls during rush-hour as police officers use hand motions at roundabouts including along the main highway that connects Kenya's Mombasa port to landlocked neighbors including Uganda. The roads are often so chaotic that police officers on foot cannot keep us in certain roads.

While some communication companies have been offering regular traffic updates on certain roads, road users would need more than just being updated whether there is traffic or not, but also how long it will take them to reach their intended destinations based on traffic conditions.

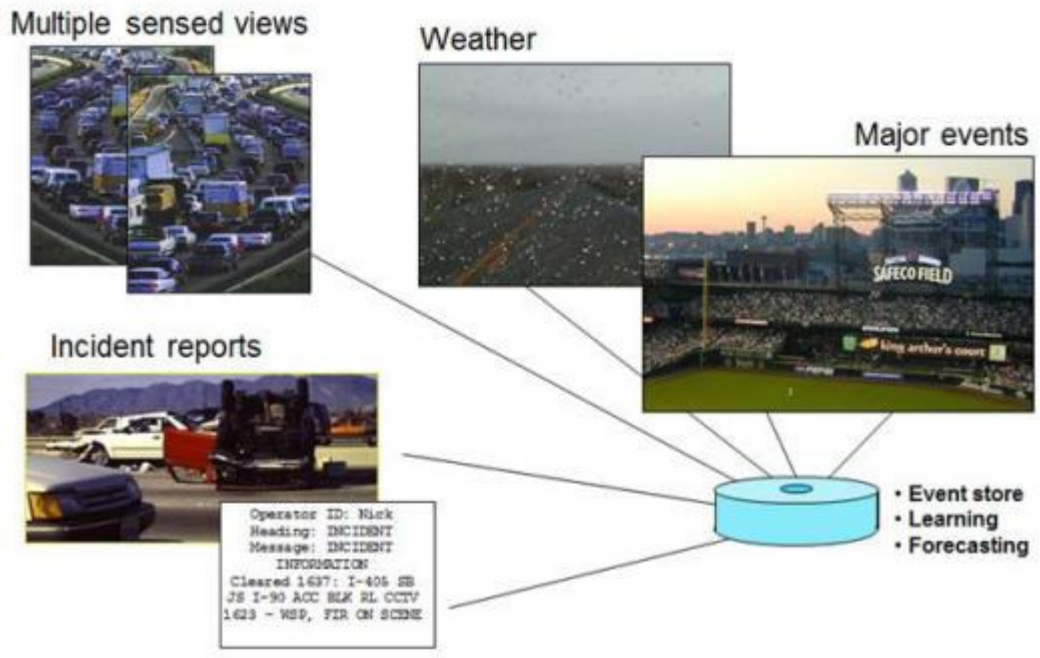
Google maps does more than provide directions, it also serves up estimated time of arrivals (ETA) by determining the time it takes to drive, walk, or bike to a destination. The ETA depends on things ranging from official speed limits and recommended speeds, likely speeds derived from road types, historical average speed data over certain time periods, actual travel times from previous users, and real-time traffic information. They mix data from whichever sources they have, and come up with the best prediction they can make (Elyse, 2013). From the above information it can concluded that several factors contribute to varying traffic patterns hence affecting arrival time of vehicles. In this research project, the proposal is to explore the use of machine learning algorithms, specifically support vector regression (SVR) to improve the accuracy of predictive models that are used to predict vehicle arrival time on a managed urban highway.

This research project mainly focuses on the development of a model that will improve the accuracy of travel time for truck drivers, using historically collected data, to provide reliable travel time information for drivers.

1.3 JamBayes Project

In 2002, JamBayes research project (Horvitz , Apacible , Sarin, & Liao, 2002) was started because of the frustrations encountered with navigating through Seattle traffic, a region that has seen great growth amidst slower changes to the highway infrastructure.

The challenge was to predict the future of traffic flow: How long would it be until a current traffic jam on the highway system of Seattle would melt? How long until open flows on different segments of the highway system of Seattle would become clogged? The idea was to combine heterogeneous streams and histories of information to make these predictions. These streams included multiple years of different types of data, including sensed highway data, reports of accidents throughout the highway system, weather, and major regional events.



```
Operator ID: Nick
Heading: INCIDENT
Message: INCIDENT
INFORMATION
Cleared 1637: I-405 SB
JS I-90 ACC BLK RL CCTV
1623 - WSP, FIR ON SCENE
```

Figure 1.3.1 JamBayes focused on learning to forecast flows from multiple streams of information

The JamBayes project employs the use of Bayesian structure search, using tools developed by Chickering and team to construct a Bayesian network (Chickering, Heckerman, & Meek, 1997). The method provides a graphical view onto the model that enables visualization of multiple variables and influences. Given a training dataset, the method performs heuristic search over a space of dependency models using a Bayesian scoring criterion to guide the search. Details on the heuristic search and model scoring can be found in (Chickering, Heckerman, & Meek, 1997). A mix of discrete and continuous variables in the models using continuous variables to represent times of different events interest is employed. For each discrete variable, the method creates a tree containing a multinomial distribution at each leaf. For each continuous variable, the method constructs a tree in which leaves contain a binary-Gaussian distribution; each leaf contains a binomial distribution that represents the probability of the value being present or absent, and a Gaussian distribution over the values that are present. The Bayesian score used for each binary-Gaussian distribution is the sum of the score for the binomial, a special case of the multinomial, and the score for a Gaussian defined over the values that are present in the training data. Rather than build separate models for each bottleneck, a large model is constructed for all bottlenecks. The comprehensive model captures the rich interdependencies among multiple bottlenecks and other variables, allowing the system to learn about dependencies and temporal relationships among the flows and congestions at bottlenecks.

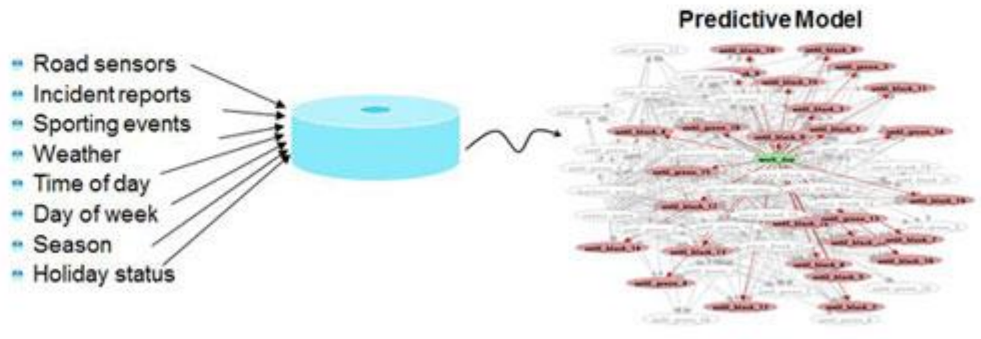


Figure 1.3.2 A representation of a predictive model

Data was analyzed and processed into a predictive model that was used in real time to generate predictions about the duration of jams on different regions of the highway system. From the study several factors that lead to varying traffic patterns along a highway which affects arrival time of vehicles are revealed. These factors includes; road sensors, incidents , events, weather, time of day, day of the week, seasons and holiday status, then modelled as set of interdependent variable to be able to generate a predictive model.

1.4 Smart Flow Prototype

The JamByes project led to the Smartflow prototype which has been used by thousands of Microsoft employees for years. Smartflow explored both the fielding of a predictive traffic service as well as navigation techniques for small devices. Microsoft announced in November 2014 that Bing maps now can not only provide live traffic updates, but with Clearflow technology predict traffic on roads that do not provide live traffic data. Bing Maps can provide more accurate trip time calculations and provide better directions/routes by using traffic predictions (Bandla, 2014). Clearflow powered prediction will now be available in every country where Bing Maps traffic coverage is live/active.

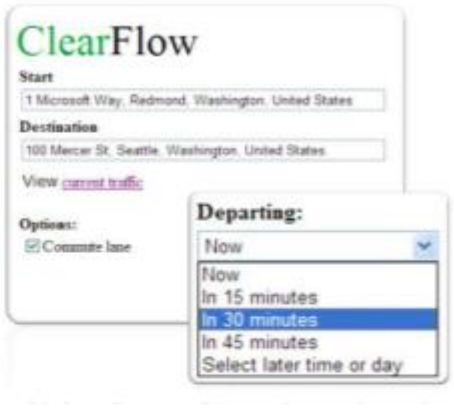


Figure 1.4.1 Input fields for Clearflow prototype fielded internally at Microsoft R&D process.

Development of a prototype to demonstrate how research results can be implemented in a commercial environment was one of the key objectives in this research study. A prototype was built from the SVR model that was designed and implemented for predictions.

1.5 Google Traffic with crowd sourcing

New generation of navigation apps on mobile platforms are crowd-sourcing GPS data from users to generate live traffic feed. Google Maps, the most popular mapping and navigation mobile app in India, has already integrated traffic feed for major Indian cities (Karthik & Srikanth, 2014). The apps determine traffic flow by plotting the movement of a navigation system user by getting details of GPS coordinates on a stretch and scanning the movement over a period of time. Google Traffic works by analyzing the GPS-determined locations transmitted to them by a large number of users. Google processes the incoming raw data about device locations, and then excludes anomalies such as a postal

vehicle which makes frequent stops. When a threshold of users in a particular area is noted, the overlay along roads and highways on the Google map changes color (Barth, 2009).
Lobbying for electronic information from a large group of people this way is referred to as crowdsourcing. Google states that when different speeds are combined across thousands of phones moving around a city at any given time, then it can get a pretty good picture of live traffic conditions.

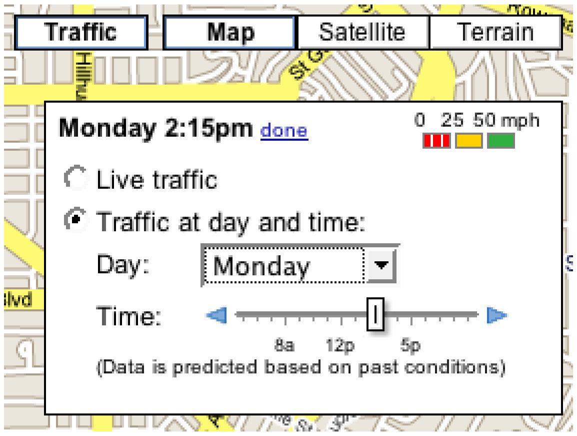


Figure 1.5.1 **Traffic prediction interface**

A point to reflect on is that many people would be concerned about telling the world how fast their car is moving if they also have to tell the world where they were going. Google has built in a number of features to safeguard the identities and locations of users who choose to provide Google with traffic data.

Mobile devices which use Google's Android operating system come equipped with the ability to send location data to Google, while non-Android devices which use Google's map application are also able to transmit their location data to Google. However, options available in each phone's "settings" allow users not to share information about their location with Google Maps (Barth, 2009). On Google's website, detailed "opt-out" instructions are available for various devices and operating systems, including Android, BlackBerry, iPhone/iPod, Palm webOS, Symbian S60, Windows Mobile, and Sony Ericsson. Google stated, "Once you disable or opt out of My Location, Maps will not continue to send radio information back to Google servers to determine your handset's approximate location".

Google has stated that the speed and location information it collects to calculate traffic conditions are anonymous. Google also identifies the start and end points of every trip it monitors, and permanently deletes that data so that information about where each user came from and went to remains private.

1.6 The Objectives of the Study

This research project sought to develop a predictive model that is to improve the accuracy of the estimated arrival time of a vehicle depending on traffic conditions. This is achieved through the analysis of data collected along the sections of a highway on specific days and time of the week. This model was then used to build a web application that can be used by the drivers to get information about the specific time they need to reach their intended destination based on the traffic condition.

The main objective of this research project was to develop and test a model that improves accuracy of prediction of the Estimated Time of Arrival (ETA) of a vehicle between two clearly defined sets of locations compared to the existing algorithms in literature. The scope of this project encompasses:

1. To collect real time data to be used to design and test different predictive algorithms.
2. Design a predictive algorithm to estimated vehicle arrival time.
3. Test and compare the accuracy of the developed model to the actual arrival time.
4. Build a prototype system that will display the estimated arrival time for driver intended source and destination based on the traffic condition on the road at the time of travel.

1.7 Significance of the Study

The full implementation of the research results into a web/mobile based application can help in building a more reliable travel time information system. This will help road users to overcome the challenge faced when a lot of time is lost in traffic due to unreliable road traffic information. This would help reduce stress levels caused to drivers hence contributing to a healthier nation. It does also allow them to make informed choices with regard to traffic conditions hence saving on cost and time which can be used for a much more productive course.

CHAPTER II

LITERATURE REVIEW

2.1 TRAVEL TIME PREDICTION MODELS

Over the years several prediction models have been developed that has led to forecasting traffic states such a traffic flow and travel time. The importance of the short-term travel time prediction has increased remarkably (Smith & Demetsky, 1995). A variety of prediction models such as historical data based models, regression models, time series models and neural network models, have been developed over the years. The five most widely used models include historical data based models (Williams & Hoel, 2003; Jeong , 2004), time series model (Thomas, Weijermars, & Van Berk, 2010; Al-Deek, H, M, & M, 1998), regression models (Jeong , 2004; Ramakrishna, Y., P. Ramakrishna, V. Lakshmanan, & R. Sivanandan, 2006), Kalman filtering model (Chien, S.I.J., Ding, Y., Wei, & C., 2002; Shalaby & Farhan, 2004) and machine learning models (Bin et al 2006, Yasdi 1999, Jeong 2004). (Kirby et al 1997, Zheng et al 2006) however, discussed that no single predictor has been developed that presents itself to be universally accepted as the best, and at all times, and effective traffic state forecasting model for real-time traffic operation. Hybrid models, e.g. a combination of Kalman filtering and neural network (Chien et al 2002, Chen et al 2004), a combination of time series and Kalman filtering (Thomas, Weijermars, & Van Berk, 2010) also drew much attention. Several classifications of prediction methods have been suggested by researchers. The focus will be given to their application in travel time prediction, mostly for transit vehicles.

2.1 Historical Data Based Models

These prediction models gives the current and future bus travel time from the historical travel time of previous journeys on the same time period. The current traffic condition is assumed to remain stationary. (Williams and Hoel 2003) pointed out that the phenomenon that traffic conditions follow nominally consistent daily and weekly patterns leads to an expectation that historical averages of the conditions at a particular time and day of the week will provide a reasonable forecast of future conditions at the same time of day and day of the week (Shalaby & Farhan, 2004). Therefore, these models are reliable only when the traffic pattern in the area of interest is relatively stable, e.g. rural areas.

2.2 Regression Models

These models predict and explain a dependent variable with a mathematical function formed by a set of independent variables (Chien et al 2002). Unlike historical data based prediction models, these are able to work satisfactorily under unstable traffic condition. Regression models usually measure the simultaneous effects of various factors, which are independent between one and another, affecting the dependent variable. (Patnaik et al 2004) proposed a set of multilinear regression models to estimate bus arrival times using the data collected by automatic passenger counter (APC).

They used distance, number of stops, dwell times, boarding and alighting passengers and weather descriptors as independent variables. They indicated that the models could be used to estimate bus arrival time at downstream stops. However, this approach is reliable when such equations can be established. (Jeong 2004) and (Ramakrishna et al 2006) also developed multilinear regression models using different sets of inputs. Both studies indicated that regression models are outperformed by other models. One great advantage of multilinear regression model is that it reveals which inputs are less or more important for prediction. For example, (Patnaik et al 2004) discovered that weather was not an important input for their model. Also (Ramakrishna et al 2006) found out that bus stop dwell times from the origin of the route to the current bus stop in minutes and intersection delays from the origin of the route to the current bus stop in minutes are less important inputs. In general, the applicability of the regression models is limited because variables in transportation systems are highly inter-correlated (Chien et al 2002).

2.3 Time Series Models

These models assume that the exogenous factors acting upon the dynamical system either remain constant, or can be measured and accounted for in the model, if they vary in time. In terms of traffic, they assume that the historical traffic patterns will remain the same in the future. As has been indicated on (Chien et al 2002), the accuracy of time series models is a function of the similarity between the real-time and historical traffic patterns. Variation in historical travel time data or changes in the relationship between historical and real-time travel time data could significantly cause inaccuracy in the prediction results. To the author’s knowledge, these models have not been used for prediction of bus travel time so far. However, they have been used and indicated to be effective for link travel time and traffic volume predictions either alone or in combination with other models, e.g. Kalman filtering (Al-Deek et al 1998, Thomas et al 2010).

2.4 Kalman Filtering Models

It has been used extensively for predicting bus arrival time (Chen et al 2004, Vanajakshi et al 2009, Wall and Dailey 1999, Chien et al 2002, Yang 2005). Chen and Chien (2002) and Wall and Dailey (1999) used Kalman filtering techniques to predict auto travel time. The Kalman filtering model has the potential to adapt to traffic fluctuation with time-dependent parameters (Chen and Chien 2002). These models are effective in predicting travel time one or two time periods ahead, but they deteriorate with multiple time steps (Park and Rilett 1999). Park and Rilett compared neural network models with other prediction models including Kalman filtering techniques to predict freeway link travel time. While the average mean absolute percentage error (MAPE) of neural network models changed from 8.7 for one time period to 16.1 for 5 time periods, that of Kalman filtering changed from 5.7 to 20.1 (Park and Rilett 1999).

2.5 Machine Learning Models

Machine learning methods present some advantages with respect to statistical methods: they are able to deal with complex relationships between predictors that can arise within large amounts of data, are able to process non-linear relationships between predictors and are able to process complex and noise data (Ricknagel 2001). These models can be used for prediction of travel time, without explicitly addressing the (physical) traffic processes (Hoogendoorn and Van Lint 2008). However, they are location-specific solutions, requiring significant efforts in input- and model selection for each specific application, via for instance correlation analysis, or genetic algorithms or trial-and-error procedures. Results obtained for one location are (typically) not transferable to the next, due to location-specific circumstances (geometry, traffic control, etc.). Artificial Neural Network (ANN) and Support Vector Regression methods are presented under these categories.

2.6 Artificial Neural Network Models

Much research has been on predicting bus arrival time using ANNs because of its ability to solve complex non-linear relationships (Chen et al 2004, Ramakrishna et al 2006, Jeong 2004, Chien et al 2002, Park et al 2004). Compared to the previously discussed models, ANNs can be developed without specifying the form of the function, while the restrictions on the multicollinearity of the explanatory variables can be neglected. Chien developed an enhanced Artificial Neural Network model to predict dynamic bus arrival time using Back-Propagation algorithm (Chien et al 2002).

On another study, Chen developed a methodology for predicting bus arrival time using data collected by Automatic Passenger Counter (APC) (Chen et al 2004).Their model consisted of an Artificial Neural Network (ANN) model to predict bus travel time between time points and a kalman filter based dynamic algorithm to adjust the predicted arrival time using bus location information. The ANN was trained with four input variables, day-of-week, time-of-day, weather and segment; and produced a baseline estimate of the travel time. The dynamic algorithm then combined the most recent information on bus location with the baseline estimate to predict arrival times at downstream time points. The algorithm not only explicitly considered variables influencing the travel time but also updated it using the real-time APC data. The authors indicated that their model was powerful in modelling variations in bus-arrival times along the service route. It was observed that the dynamic algorithm performed better than the corresponding ANN model because it incorporated the latest bus-arrival information into the prediction. The ANN model also performed better than the timetable. (Jeong and Rilett 2004) also proposed an ANN model for predicting bus arrival times and demonstrated its superior performance as compared with the historical data based and multi-linear regression models. Historical data based model gave superior results, as compared to the multiple linear regression. The authors have tested 12 training and 14 learning functions and the best functions were chosen for the prediction purpose.

The advantage of their models was that traffic congestion, schedule adherence and dwell times at stops were considered as inputs for the prediction. (Ramakrishna et al 2006) developed a Multiple Linear Regression (MLR) model and an Artificial Neural Network (ANN) model for prediction of bus travel times using GPS-based data. These models were applied to a case study bus route in Chennai city, India. It was indicated that Artificial Neural Network model performed better than Multiple Linear Regression model.

In general, ANN models have the ability to capture the complex non-linear relationship between travel time and the independent variables. These models have been proved to be effective for the provision of satisfactory bus arrival time information. They could be very useful in prediction when it is difficult or even impossible to mathematically formulate the relationship between the input and output. Though the learning and testing process is inherently delicate and is slow to converge to the optimal solution (Hagal et al 1996), it is still possible to do an off-line training and adapting ANNs to real-time condition if the inputs are chosen carefully.

2.7 Support Vector Machines

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. While other machine learning techniques, such as ANN, have been extensively studied, the reported applications of SVM in the field of transportation engineering are very few. Support vector machine and support vector regression (SVR) have demonstrated their success in time-series analysis and statistical learning (Chun-Hsin et al 2003). Since support vector machines have greater generalization ability and guarantee global minima for given training data, it is believed that support vector regression will perform well for time series analysis. (Bin et al 2006) proposed Support Vector Machine (SVM) as a new neural network algorithm to predict the bus arrival time. They pointed out that unlike the traditional ANN, SVM is not amenable to the over fitting problem, and it could be trained through a linear optimization process. This study predicted the arrival time based on the travel time of a current segment and the latest travel time of the next segment. The authors built separate models according to the time-of-day and weather conditions.
The developed model was tested using off-line data of a transit route and exhibited advantages over an ANN based model methods. These models have been developed for prediction of travel time on highways, e.g. (Chun-Hsin et al 2003). They compared their proposed SVR predictor to other baseline predictors, the results showed that the SVR predictor can reduce significantly both relative mean errors and root mean squared errors of predicted travel times. However, (Bin et al 2006) indicated that when SVM is applied for solving large-size problems, a large amount of computation time will be involved. In addition, the methods for selecting input variables and identifying the parameters should be further researched.

2.8 Travel time distribution

Travel time distribution is a result of the fluctuations in traffic demand and supply (Van Lint et al 2003, (Tu, 2008). Travel times are the result of traffic flow operations, which in turn are attributed to the interaction between traffic demand and traffic supply characteristics. These characteristics are categorized as below:

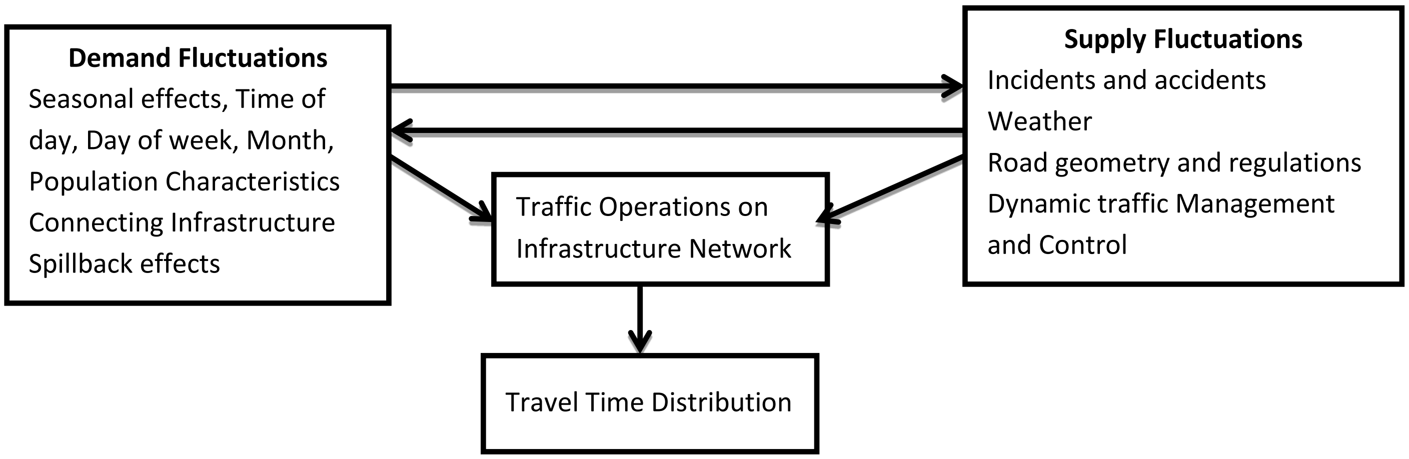

Demand Fluctuations
Seasonal effects, Time of day, Day of week, Month, Population Characteristics
Connecting Infrastructure
Spillback effects
Supply Fluctuations
Incidents and accidents
Weather
Road geometry and regulations
Dynamic traffic Management and Control
Traffic Operations on Infrastructure Network
Travel Time Distribution

Figure 2.8.1 Overview of factors influencing the distribution of travel times (Tu 2008, (Zegeye & Nall, 2014)

To improve the reliability of traffic information and increase the accuracy of travel time predictions, the knowledge of travel time variability is valuable. Therefore, a reduction in travel time variability reduces the uncertainty in decision-making about departure time and route choice as well as the anxiety and stress caused by such uncertainties. As far as these fluctuations continue to exist, one will frequently experience variability in travel time irrespective of the type of vehicle. Predicting travel time along certain roadways or routes can be challenging due high degree of variability which in turn may result in less reliable travel time information.

Very few studies have concentrated on quantifying sources of uncertainties making travel time unreliable. Asakura has categorized the sources of travel time fluctuations in to three factors which are from demand side such as day to day traffic variation, supply side such as road closure due to accidents and external factors such as adverse weather effects and natural disaster (Asakura 2006). Most of the studies in the literature used deterministic approach to model travel time variation under the influence of various factors from supply side and demand side of the system.

Ruimin examined travel time variability under the influence of time of day, day of week, weather effect and traffic accident (Ruimin, 2004). In that study, the author quantified sources of travel time parameters with the help of multiple linear regressions with two way interaction models. The results of his study is summarized in the table below

	Independent Variables	Adjusted R Square
Morning peak	Day of week, time of day, weather, incident and all interaction terms	0.56
	Time of day, day of week, weather, incident	0.55
	Time of day, day of week	0.53
Afternoon peak	Day of week, time of day, weather, incident and all interaction terms	0.50
	Time of day, day of week, weather, incident	0.43
	Time of day, day of week	0.25

Ruimin thus concluded from the above findings that day of week and time of day effects have more significant influence on morning peak travel times than on afternoon peak travel times. Whereas effects like rain and incidents, contribute more variability to the afternoon travel times than the morning travel times. Therefore, almost half travel time variability in morning peak is caused by demand related variations, while the 25% of the travel time variability in afternoon peak results from the capacity related variations. This is consistent with the traffic situation of the study site.

Ruimin however stated that his research was more focused on the day-to-day instead of time-to-time travel time variability thus to mean that the day-to-day travel time variability measures the change of travel times for the same trip at the same time from day-today, while time-to-time travel time variability emphasizes the change of travel times of the same trip on the same day from time to time. This study has further recommended that travel time data across a longer time period than the one-month used are needed to separate out the effect of time of day and day of week.

In another study, Van identified the main causes of unreliability of travel times for Netherlands urban roads. According to his study, 74 % of unreliability in the travel time is mainly due to internal factors of the traffic. The remaining is due to weather (8 %), road works (14 %), accidents (3 to 12 %) and combination factors (2 %) (Van Zuylen, 2004).

From Tu's research recommendations this research proposed a travel time distribution model that is suitable under the following conditions:-

I. The model is suited for all motorways under all conditions,
II. Modeling the reliability of individual travel times, using monitoring data from single vehicles,
III. Modeling reliability of travel times on urban road networks. (Tu, 2008)

The proposed SVR travel time conceptual framework for this research is shown in the diagram below:

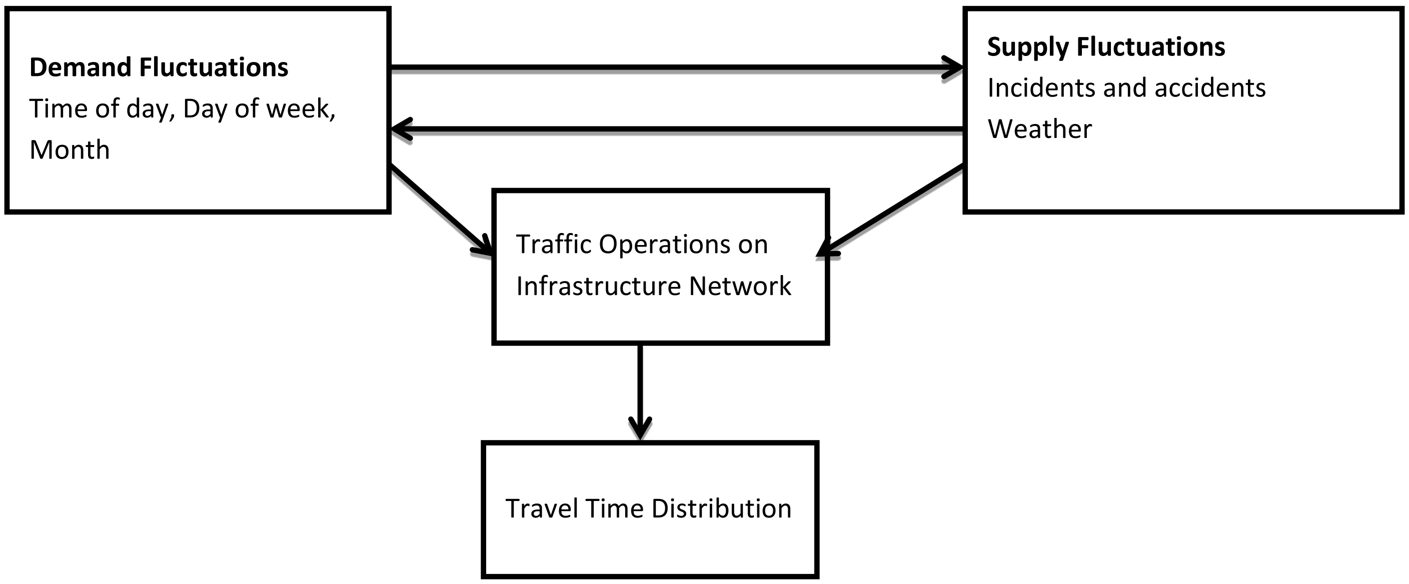


Figure 2.8.2 The SVR based conceptual framework for travel distribution model

The travel time distribution framework has been reduced to reflect only the factors that the research focused on or have significant effect on travel time during the period of study. The rest of the factors are treated as constants and data collected under the influence of those factors were not considered in developing the prediction model algorithm. This model can be justified by the fact that the population characteristics, seasonal effects, cultural factors, connecting infrastructure and spillbacks a part of the demand fluctuations as indicated in Tu's model can be treated as impact variables and in most cases they largely remained constant during the period and section of study. Under supply fluctuations, road geometry and dynamic traffic management and control as constant as well were treated as constant.

This research though recommends that future include as many variables from Tu's model and model those variables into mathematical quantities that SVR models accept. Future research could involve studying the impact of these factors on travel time distribution models as well.

2.9 Summary

This chapter presented a literature review on travel time prediction models. According to this literature review, the provision of accurate and timely arrival time information has been widely researched using different approaches. To predict travel time, many models have been developed including historical data based models, regression models, time series models, and artificial neural network models. Machine learning methods have also been applied to create models that provide estimates of flow inferences about current and future traffic flows. However, while there is still more research options in the use of support vector machines in travel time predictions.

Travel time information can be theoretically achieved by investigating the impacts of each the mentioned above on the distribution of travel time. But due to limited resources in developing countries, it is not easy to get information or data to study the impacts of all those factors. Therefore, prediction of travel time needs to be done using the available data along with logical assumptions.

CHAPTER III

RESEARCH METHODOLOGY

3.1 Introduction

The previous chapter provided a review of literature in three areas related to this study: the factors that contribute to varying traffic conditions and their contribution to this study; the different approaches that have been used to develop arrival time prediction models and analyzed both their weaknesses and strengths; examined the factors that have greater impact to vehicle arrival time; then proposed SVR conceptual framework that will be used to develop a time prediction model for this study. This review of literature provided guidance in the development of this study.

This chapter discussed the approach used to develop and validate the arrival time prediction model for trucks; it describes the research design, the case study, instrumentation, data collection, and data analyses that were used in the study.

The purpose of this study was to design an SVR based model that predicts vehicle arrival time between two clearly defined sets of road locations. Taking into account the movements and stops of the trucks, the mechanism of travel time prediction proposed in this paper uses data mobility-coordinates with timestamp to "learn" then estimate the time a truck will arrive at a given predefined location given the time and place of departure. This study uses experimental investigation approach together with simulation techniques to achieve its research purpose.

3.2 Research Methods

The study used mixed methods, incorporating the case study, exploratory, agile, analysis and simulation methodologies to achieve the research objectives.

This research used a case study research design to perform data collection procedures along Uhuru Highway, Nairobi Kenya. The trucks passing along that route was selected as the unit of analysis. Exploratory study was used when designing the SVR prediction model. Simulation method was used when tuning the SVR parameters before the performance of the developed models are evaluated. Analysis methodology using statistical methods was used to evaluate and compare the performance of the developed models. The agile method was used when developing the prototype that as proposed in the research objective.

The four purposes of this chapter were to use the proposed methodologies to perform the following task.

1. Collection and processing of raw data to create aggregated data.
2. Develop and evaluated a model to learn the behavior of the trucks using this aggregated data.
3. Predict the estimated arrival time based on the developed model
4. Build a prototype to demonstrate how the results can be implemented to a working time prediction system

The following research framework is proposed under this topic.

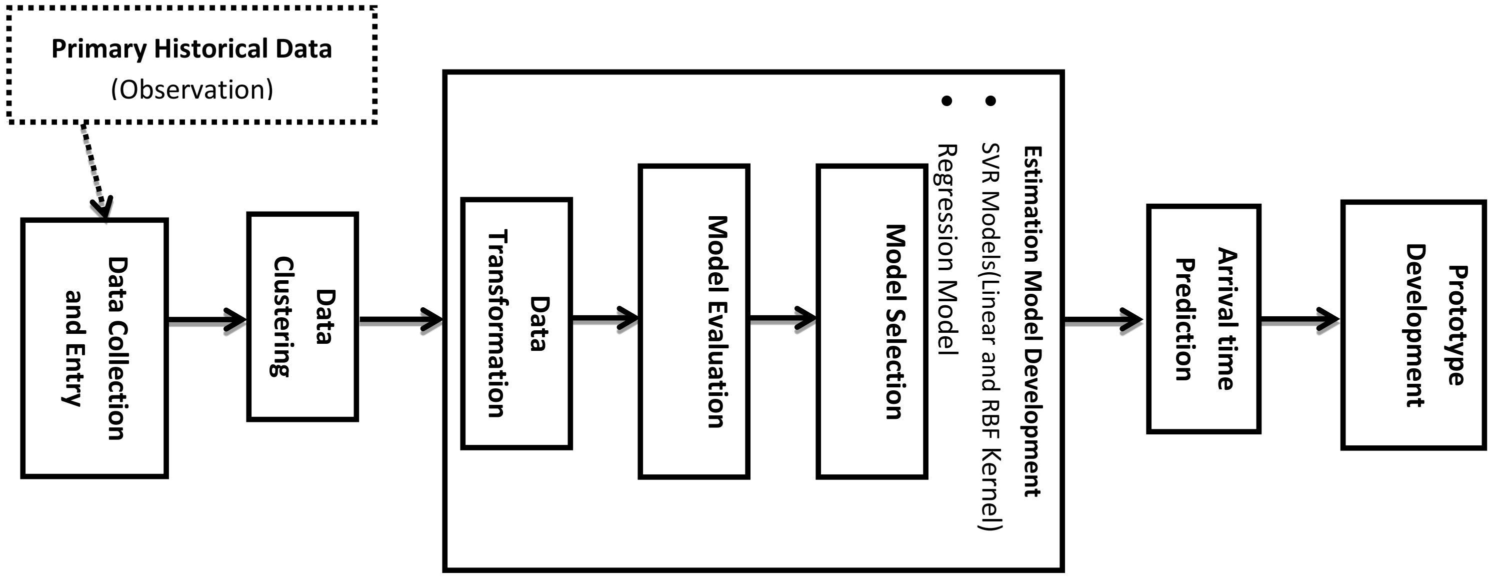


Figure 3.1.1 Research Framework

3.3 Data Collection and Entry

3.3.1 Method

Covert observation was used as a method of collecting raw data for this research. Observation is defined as a way of gathering data by watching behavior, events, or noting physical characteristics in their natural setting. The benefit of covert observation is that people are more likely to behave naturally if they do not know they are being observed. However, you will typically need to conduct overt observations because of ethical problems related to concealing your observation.

Observations can also be either direct or indirect. Direct observation is when you watch interactions, processes, or behaviors as they occur. Indirect observations are when you watch the results of interactions, processes, or behaviors. The advantages of observation as a method of data collection is because of the following:-

i. Directness

The main strength of observation is that it provides direct access to the event under consideration. Instead of relying on some kind of self-report, such as asking people or secondary data, you actually observe and record their behavior in that situation. This, in principle at least, avoids the wide range of problems associated with self-report.

ii. Diversity, Flexibility and Applicability

Observation can take diverse forms, from informal and unstructured approaches through to tightly structured, standardized procedures and can yield associated diverse types of data, both qualitative and quantitative. Observation, therefore, is applicable in a wide range of contexts.

iii. Provision of a permanent record

The fact that all observation entails some form of recording means that it provides a permanent record of such events or behavior, thus allowing further analysis or subsequent comparisons across time or location to be carried out.

iv. Complementarity with other approaches

Using more than one technique of data collection through a process of triangulation is seen as highly desirable as an overarching research strategy. Therefore, strength of observation is that it can effectively complement other approaches and thus enhance the quality of evidence available to the researcher.

3.3.2 Data Acquisition

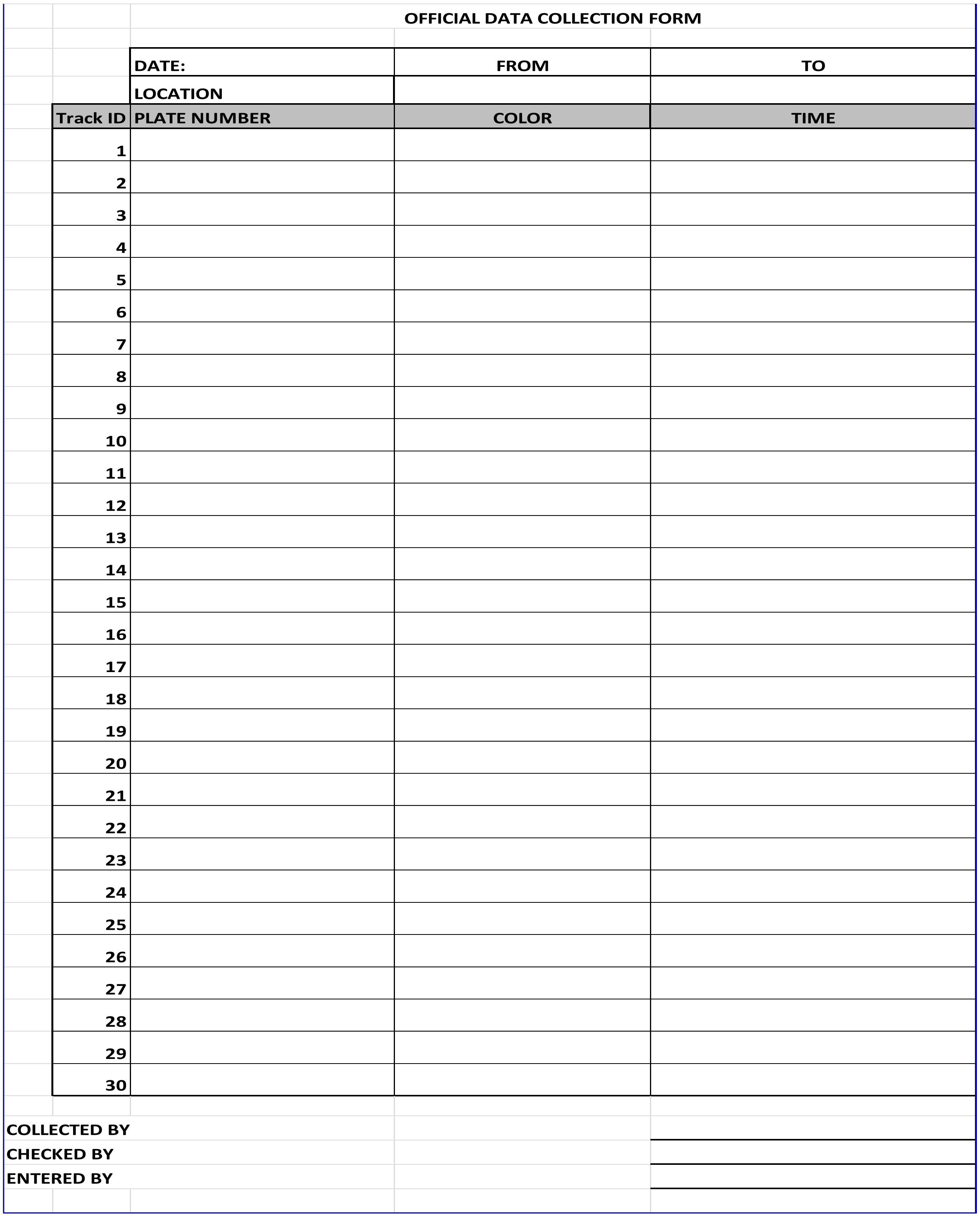
Data clerks were selected from a list of successful candidates who were shortlisted for the interviews for the task of data collection and entry. The interview guide is attached in **Appendix C**. Since the exercise involved the need to collect primary data, the interview focused on their ability to collect data with highest level of integrity and ability to endure long periods collecting data with similar characteristics. The data collection materials were availed to them during the interview process. Training on data collection techniques via observation method was done on the first day of data collection and it continued part of the on the job training conducted after every one week. The training and test data was collected using a set of pre-designed forms. The data collection tool is attached in **Appendix C**. The data collection form required the clerks to capture their point locations by name. They were also required to capture trucks plate number, head color, , date and time of departure, day of the week accidents and incidents, weather conditions, police presence.

The use of these variables was the key for this work, because it was important to know when and where the vehicle is. These values could help to accurately predict the time the truck will spend to move within a particular section within the case study area. For example, if the current time is within peak hours, the truck may spend more time to move within a particular section than during off peak time for the same place. The same section could have different estimated times depending on the truck, driver and current section. Some drivers can be faster than others, or some vehicles can be older and slower than others. For each section, a model was created with these variables in mind.

Data was collected from 8th September 2014 to 22nd September 2014. The details of approximately 4,000 trucks that passed between 8 a.m. to 5 p.m. from Nyayo Stadium to Westlands roundabout during the mentioned period were captured.

3.3.3 Storage and Availability

The physical forms were collected at the end of the day and grouped per location per day and stored safely in storage box to avoid data loss or record tampering. The physical data records were then transferred into a spreadsheet for cleansing in preparation for analysis. Both the physical and electronic records are available upon request through the authority of the lead researcher. The data collection forms had the following details: date of the month, vehicle plate number and color of truck, time of exit from a section, section of study, weather and name of data clerk. A sample form is attached below:



OFFICIAL DATA COLLECTION FORM

	DATE:	FROM	TO
	LOCATION		
Track ID	PLATE NUMBER	COLOR	TIME
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			

COLLECTED BY

CHECKED BY

ENTERED BY

Figure 3.3.3.1 Sample Data collection form used

3.4 Case Study

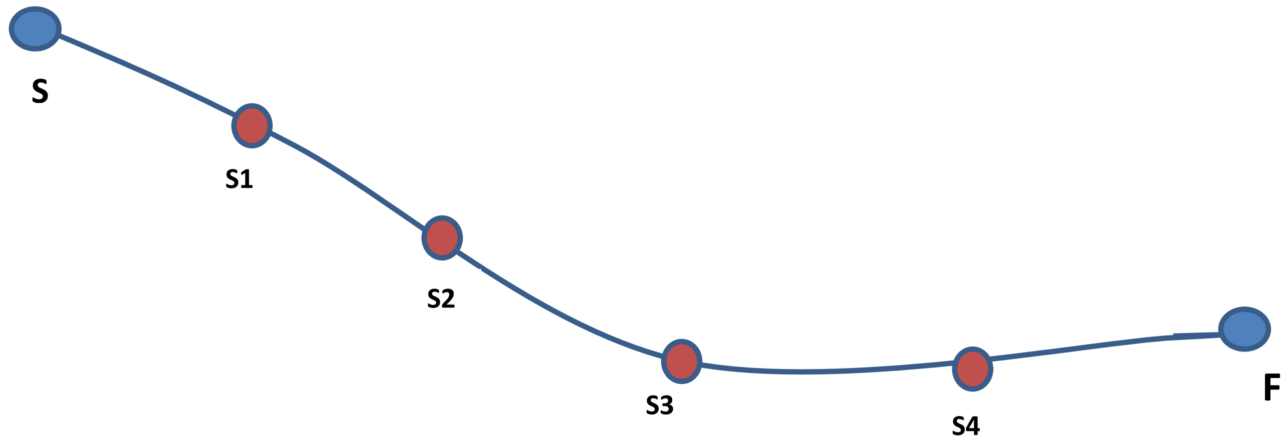
A case study usually involves a detailed study of a particular case. The method used for data collection was observation and the reasons for this choice were discussed in the previous section. Case studies have a very narrow focus which results in detailed descriptive data which is unique to the case(s) studied. This case study involves the study of arrival time of trucks that passes along Nyayo stadium round about to Westlands roundabout during the day between 8 a.m. to 5 p.m.

The main purpose of this case study is to collect data to develop an SVR based arrival time prediction model. The model is then analyzed and compared to the existing models to examine whether SVR model improves the accuracy in predicting truck arrival time. The results can then be adopted in developing future arrival time prediction systems.

3.4.1 Unit of Analysis

The unit of analysis is the major entity that is being analyzed in a case study. For this study, the unit of analysis is the trucks that pass along the sections of the road covering the Westlands roundabout to Nyayo Stadium roundabout. Data describing the characteristics of a truck such as its positon, color place and time is collected. The route was chosen because it is one of the most challenging routes in terms of traffic congestion in controlled urban traffic roads. Trucks were chosen because they provided this study with more reliable data since they can be easily traced as well as likely to move all the way from source to destination of interest.

Figure 3.4.1.1: A Section of the Road under study.



A Section of the Road under study

1. **F-**Westland Roundabout
2. **X4**-University of Nairobi Roundabout
3. **X3**-Uhuru Highway and Kenyatta University Roundabout
4. **X2**- Haile Selassie Roundabout
5. **X1**-Bunyalla Roundabout
6. **S**-Nyayo Roundabout

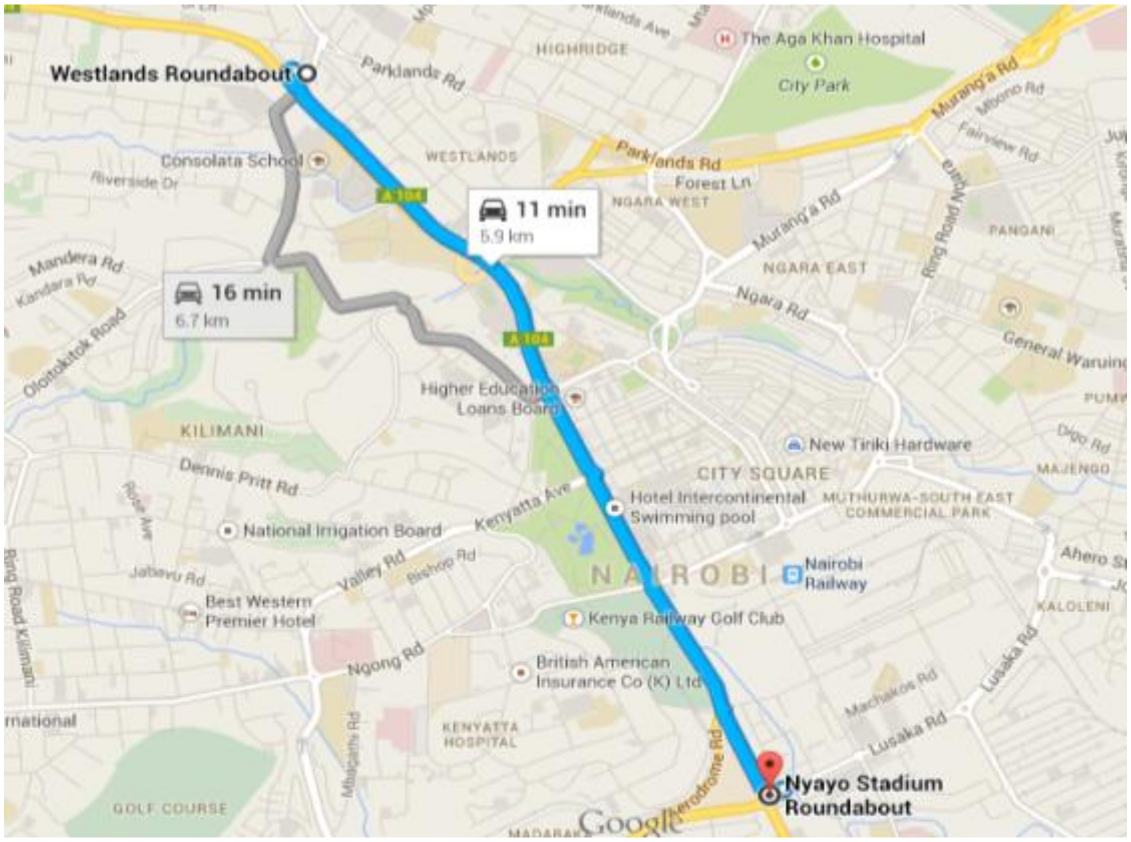


Figure 3.4.1.2: Google map representation of the road section of study

The mechanism of estimated travel time is based on the recorded movements of trucks over time, as the goal is to estimate the travel time of a specific truck between two road segments. Thus, it is necessary to know when and where the truck has moved and when and where it has stopped but more specifically at the designated sections of study. This is achieved by pre-processing the collected data to create more quantitative information.

3.4.2 Traffic affected segment processing

There can be a variation of truck speed on the road between two stops of a truck, causing some parts to be slower than others. Segmenting portions of the movement of trucks allows us to represent this difference in speed over a movement. This segmentation allows the study to accurately know how much time the truck will spend from its current position to the next known location (based on the application context) it will stop considering portions of the segment with different average speed. The stretch is done in sections as illustrated above in figure 3.4.1.1.

Considering the movement between source and destination **S** and **F** respectively, four sections points are defined: **X1, X2, X3** and **X4**. Portion **S1** starts at the stop **F**. Portion **X2** starts at the position indicated in the figure and ends at stop F. Portion **S3** starts at the position indicated in the figure and ends at stop **F**. And finally, portion **X4** starts at the position indicated in the figure and ends at stop **F**. Another way to portion the road sections would be as follows: the first portion would begin at **S** and end at **X2**. The second portion would begin at **X2** and end at **X3**. The third portion would begin at **X3** and end at **X4**. And finally the last portion would begin at **X4** and end at **F**. The research used this option to avoid complexity at data collection points that might introduce errors at that stage. The segmentation of movements creates a new entity in the model illustrated in figure 3.4.1.1. Thus, a movement (Move) has a set of entities Segment. The entity Segment has as attribute: the start point (or location), the end point (or location), the start timestamp, the end timestamp and the distance traveled.

3.5 Estimation Model Development

3.5.1 Support Vector Machines (SVM)

Support vector machines (SVM) are a group of supervised learning methods that can be applied to classification or regression. Support vector machines represent an extension to nonlinear models of the generalized portrait algorithm developed by Vladimir Vapnik. The SVM algorithm is based on the statistical learning theory and the Vapnik-Chervonenkis (VC) dimension introduced by Vladimir Vapnik and Alexey Chervonenkis. The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes.

In classification, Support Vector Machine (SVM) operates by finding the hyper plane that maximizes the margin between the two classes. The vectors (cases) that define the hyper plane are the support vectors (Sayad, 2010).

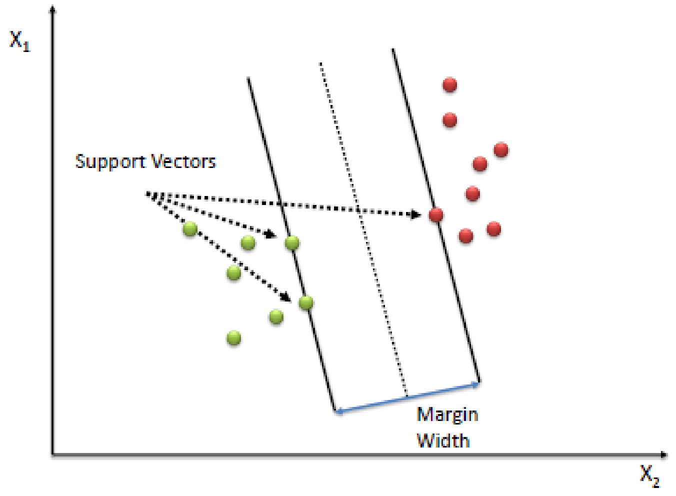
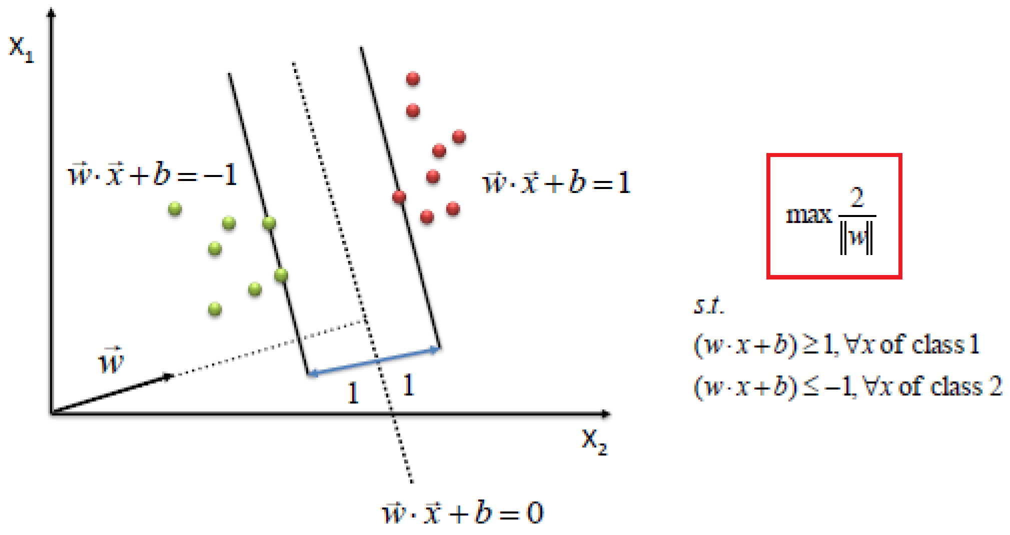


Figure 3.5.1.1 Support vector representation

Algorithm

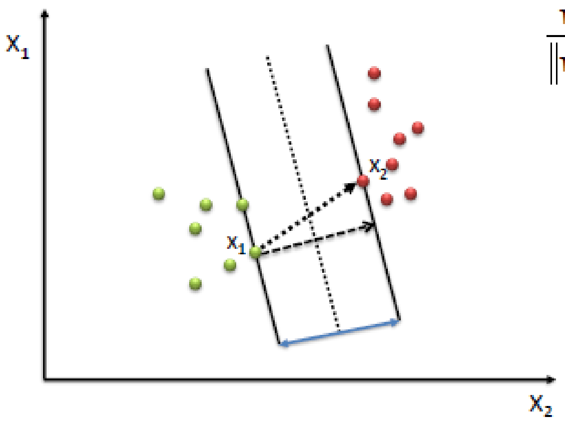
1. Define an optimal hyper plane: maximize margin
2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications.
3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space.

To define an optimal hyper plane there is need to maximize the width of the margin (w).



X_1
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\max \frac{2}{\|w\|}$
s.t.
$(w\cdot x+b)\geq 1, \forall x \text{ of class } 1$
$(w\cdot x+b)\leq -1, \forall x \text{ of class } 2$
$\vec{w}$
1
1
X_2
$\vec{w}\cdot\vec{x}+b=0$

Figure 3.5.1.2 Support vector hyper plane



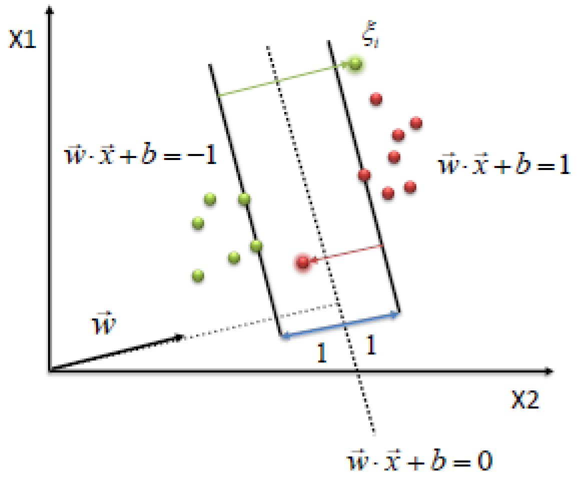
$$\frac{w}{\|w\|}\cdot(x_2 - x_1) = \text{width} = \frac{2}{\|w\|}$$

$$w \cdot x_2 + b = 1$$
$$w \cdot x_1 + b = -1$$
$$w \cdot x_2 + b - w \cdot x_1 - b = 1 - (-1)$$
$$w \cdot x_2 - w \cdot x_1 = 2$$
$$\frac{w}{\|w\|}(x_2 - x_1) = \frac{2}{\|w\|}$$

We find w and b by solving the following objective function using Quadratic Programming.

$$\min \frac{1}{2}\|w\|^2$$
$$s.t.\ y_i(w \cdot x_i + b) \geq 1,\ \forall x_i$$

For linearly separable data, there is a unique global minimum value. An ideal SVM analysis should produce a hyper plane that completely separates the vectors (cases) into two non-overlapping classes. However, perfect separation may not be possible, or it may result in a model with so many cases that the model does not classify correctly. In this situation SVM finds the hyper plane that maximizes the margin and minimizes the misclassifications.



X1
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X2
$\vec{w}\cdot\vec{x}+b=0$

slack variable:

ξ_i

Allow some instances to fall off the margin, but penalize them

Figure 3.5.1.3: Slack variable in SVM

The algorithm tries to maintain the slack variable to zero while maximizing margin. However, it does not minimize the number of misclassifications but the sum of distances from the margin hyper planes.

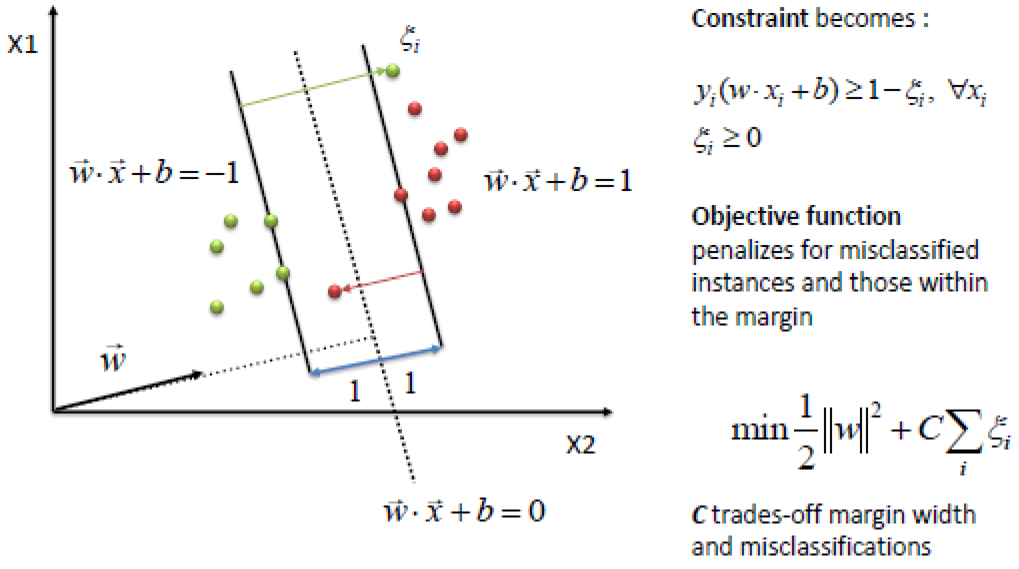
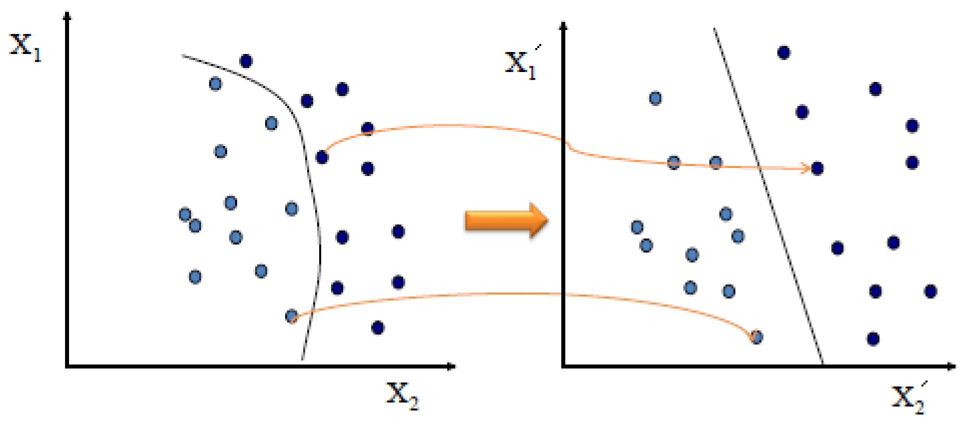
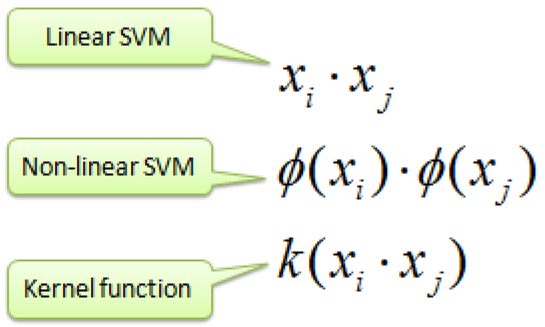

X1
X2
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}$
1
1
Constraint becomes :
$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$
$\xi_i \geq 0$
Objective function penalizes for misclassified instances and those within the margin
$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$
C trades-off margin width and misclassifications

Figure 3.5.1.4: SVR diagram

The simplest way to separate two groups of data is with a straight line (1 dimension), flat plane (2 dimensions) or an N-dimensional hyperplane. However, there are situations where a nonlinear region can separate the groups more efficiently. SVM handles this by using a kernel function (nonlinear) to map the data into a different space where a hyperplane (linear) cannot be used to do the separation. It means a non-linear function is learned by a linear learning machine in a high-dimensional feature space while the capacity of the system is controlled by a parameter that does not depend on the dimensionality of the space. This is called kernel trick which means the kernel function transform the data into a higher dimensional feature space to make it possible to perform the linear separation (Sayad, 2010).





Map data into new space, and then take the inner product of the new vectors. The image of the inner product of the data is the inner product of the images of the data. Two kernel functions are shown below.

Polynomial

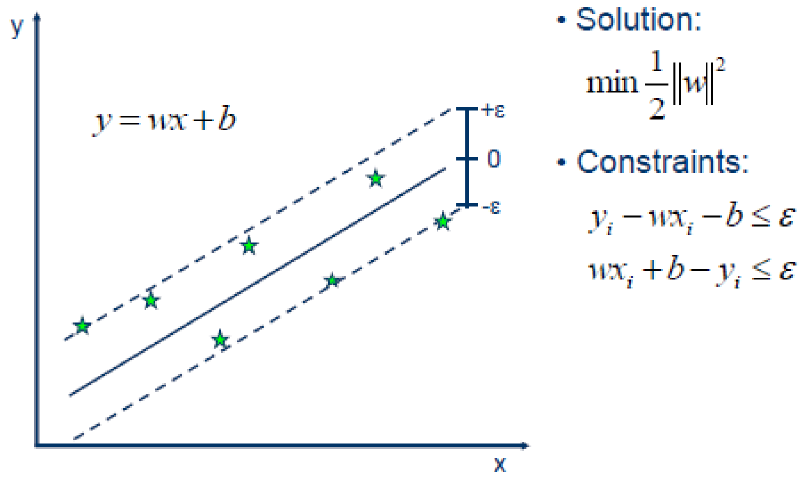
$$k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i . \mathrm{x}_j)^d$$

Gaussian Radial Basis function

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\left\| \mathrm{x}_i - \mathrm{x}_j \right\|^2}{2\sigma^2} \right)$$

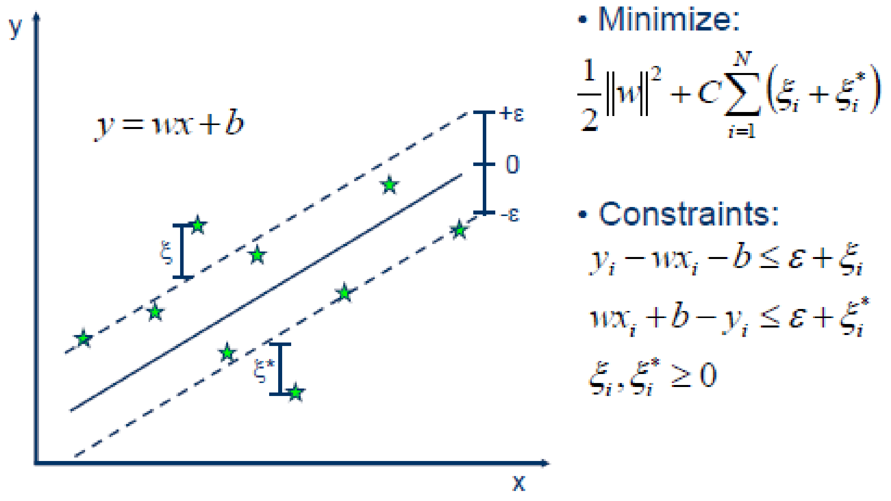
3.5.2 Support Vector Machine - Regression (SVR)

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences (Sayad, 2010). First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason; the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyper plane which maximizes the margin, keeping in mind that part of the error is tolerated.



y
x
$y = wx + b$
+ε
0
-ε
• Solution:
$\min \frac{1}{2}\|w\|^2$
• Constraints:
$y_i - wx_i - b \le \varepsilon$
$wx_i + b - y_i \le \varepsilon$

Figure 3.5.2.1 Support vector regression



y
x
$y = wx + b$
+ε
0
-ε
ξ
ξ*
• Minimize:
$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$
• Constraints:
$y_i - wx_i - b \le \varepsilon + \xi_i$
$wx_i + b - y_i \le \varepsilon + \xi_i^*$
$\xi_i, \xi_i^* \ge 0$

Linear SVR

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle x_i, x \rangle + b$$

Non-linear SVR

The kernel functions transform the data into a higher dimensional feature space to make it possible to perform the linear separation.

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle \varphi(x_i), \varphi(x) \rangle + b$$

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot K(x_i, x) + b$$

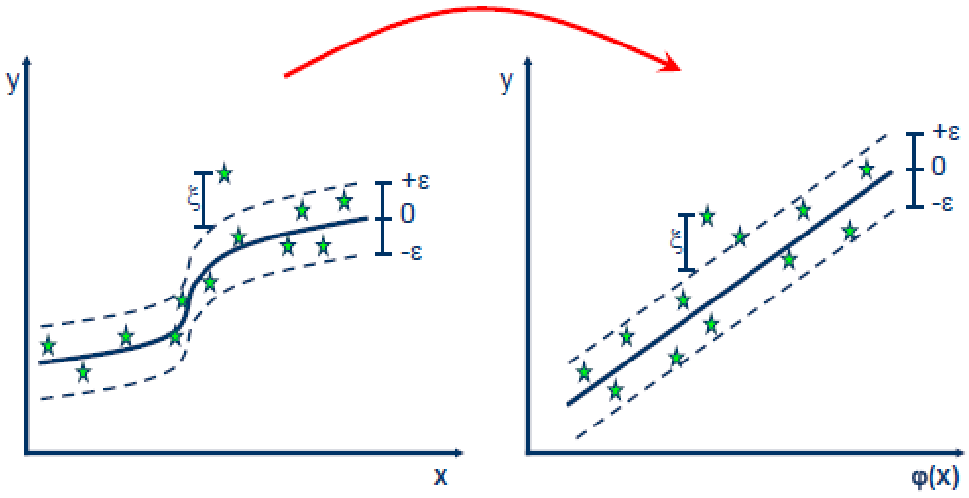


Figure 3.5.2.2: Linear Kernel for SVR

Kernel functions

Polynomial

$$k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i . \mathrm{x}_j)^d$$

Gaussian Radial Basis function

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(- \frac{\|\mathrm{x}_i - \mathrm{x}_j\|^2}{2\sigma^2} \right)$$

It is well known that SVM generalization performance (estimation accuracy) depends on a good setting of meta-parameters parameters C, £ and the kernel parameters. The problem of optimal parameter selection is further complicated by the fact that SVM model complexity (and hence its generalization performance) depends on all three parameters. Existing software implementations of SVM regression usually treat SVM meta-parameters as user-defined inputs. Selecting a particular kernel type and kernel function parameters is usually based on application-domain knowledge and also should reflect distribution of input (x) values of the training data.

3.6 Designing the model

3.6.1 Procedure for designing the model

To create the time estimation model, Support Vector Regression (SVR) was adopted. SVR uses the concept of supervised learning, in which a given set of input data, where each entry has the value to be estimated and the values of the variables of the model, "learns" the behavior of the data through the training and testing process. SVR defines a weight for each variable. The training process changes the values of these weights so that the value estimated by the model approaches the estimated value of the entry, and the testing process shows how the model "learns" the behavior of the data that were used in training process. According to (Chun-Hsin et al 2003), for this application, SVR has better results than Artificial Neural Networks because SVR is more amenable to generalization than Artificial Neural Networks. The following procedure was followed when developing the base SVR model

The procedure and the context model were employed in order to find the best prediction model parameters that were used in choosing the best arrival time prediction algorithm.

1. Transform data to the format of an SVM package
2. Consider the use of RBF kernel as the base model
3. Conduct simple scaling on the data
4. Use cross-validation and Grid Search to find the best parameter C and gamma-r
5. Use the best parameter C and gamma-r to train the whole training set
6. Test the generated model for accuracy

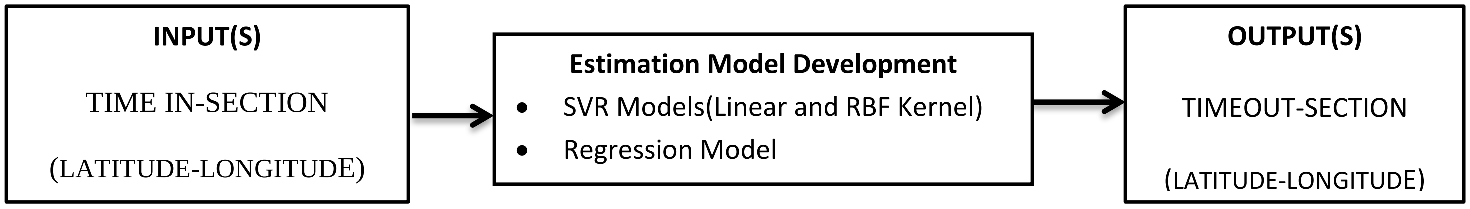


Figure 3.6.1.1: The estimated arrival time context model

3.6.2 Transform data to the format of an SVM package

SVM requires that each data instance is represented as a vector of real numbers. Hence, if there are categorical attributes, that has to be converted into numeric data. Time as recorded in the data collection form and entered in an excel spreadsheet must be converted into a set of real numbers. The spreadsheet that was used however has an inbuilt feature that automatically transforms time data in a set of unique real numbers. This made it the ideal spreadsheet for holding the data that was used for this entire data analysis process. The two tables below shows the difference between the raw data and transformed data.

No.	Date	Day	Plate	Nyayo	Bunyala	HaileS	Nyayo Hse	University	Westlands
1	8-Sep-14	Monday	KBX 192X	8:37:00 AM	8:43:00 AM	8:51:00 AM	8:58:00 AM	9:05:00 AM	9:13:00 AM
2	8-Sep-14	Monday	KBD 351F	8:43:00 AM	8:51:00 AM	8:58:00 AM	9:05:00 AM	9:13:00 AM	9:20:00 AM
3	8-Sep-14	Monday	KBL 462R	8:47:00 AM	8:55:00 AM	9:03:00 AM	9:11:00 AM	9:19:00 AM	9:25:00 AM
4	8-Sep-14	Monday	KBT 243B	8:52:00 AM	9:00:00 AM	9:08:00 AM	9:15:00 AM	9:23:00 AM	9:31:00 AM
5	8-Sep-14	Monday	KBJ 290R	9:08:00 AM	9:15:00 AM	9:24:00 AM	9:33:00 AM	9:41:00 AM	9:49:00 AM
6	8-Sep-14	Monday	KAR 936H	9:12:00 AM	9:22:00 AM	9:31:00 AM	9:39:00 AM	9:47:00 AM	9:58:00 AM
7	8-Sep-14	Monday	KBR 975R	9:13:00 AM	9:21:00 AM	9:29:00 AM	9:35:00 AM	9:40:00 AM	9:49:00 AM
8	8-Sep-14	Monday	KBZ 499E	9:13:00 AM	9:22:00 AM	9:31:00 AM	9:38:00 AM	9:42:00 AM	9:47:00 AM
9	8-Sep-14	Monday	KBW 130D	9:17:00 AM	9:25:00 AM	9:33:00 AM	9:40:00 AM	9:48:00 AM	9:57:00 AM
10	8-Sep-14	Monday	KBY 416A	9:20:00 AM	9:27:00 AM	9:35:00 AM	9:42:00 AM	9:49:00 AM	9:56:00 AM

Table 3.6.2.1 Processed raw data recorded as observed

No.	Date	Day	Plate	Nyayo	Bunyala	HaileS	Nyayo Hse	University	Westlands
1	8-Sep-14	Monday	KBX 192X	0.359028	0.363194	0.368750	0.373611	0.378472	0.384028
2	8-Sep-14	Monday	KBD 351F	0.363194	0.368750	0.373611	0.378472	0.384028	0.388889
3	8-Sep-14	Monday	KBL 462R	0.365972	0.371528	0.377083	0.382639	0.388194	0.392361
4	8-Sep-14	Monday	KBT 243B	0.369444	0.375000	0.380556	0.385417	0.390972	0.396528
5	8-Sep-14	Monday	KBJ 290R	0.380556	0.385417	0.391667	0.397917	0.403472	0.409028
6	8-Sep-14	Monday	KAR 936H	0.383333	0.390278	0.396528	0.402083	0.407639	0.415278
7	8-Sep-14	Monday	KBR 975R	0.384028	0.389583	0.395139	0.399306	0.402778	0.409028
8	8-Sep-14	Monday	KBZ 499E	0.384028	0.390278	0.396528	0.401389	0.404167	0.407639
9	8-Sep-14	Monday	KBW 130D	0.386806	0.392361	0.397917	0.402778	0.408333	0.414583
10	8-Sep-14	Monday	KBY 416A	0.388889	0.393750	0.399306	0.404167	0.409028	0.413889

Table 3.6.2.2 Transformed data from the processed raw data

3.6.3 Base parameter selection

Though there are only four known common SVM kernels, it had to be decided which one to try first. The RBF Kernel was chosen and then the penalty parameter **C** and **gamma-r** kernel parameters were arbitrarily picked.

In general, the RBF kernel is a reasonable first choice (Chih, Chang, & Lin, 2010). This kernel nonlinearly maps samples into a higher dimensional space so it, unlike the linear kernel, can handle the case when the relation between class labels and attributes is nonlinear. Furthermore, the linear kernel is a special case of RBF Keerthi and Lin (2003) since the linear kernel with a penalty parameter ~ C has the same performance as the RBF kernel with some parameters (C: r). In addition, the sigmoid kernel behaves like RBF for certain parameters (Lin and Lin, 2003).

The second reason is the number of hyper parameters which influences the complexity of model selection. The polynomial kernel has more hyper parameters than the RBF kernel. Finally, the RBF kernel has fewer numerical difficulties. Moreover, we must note that the sigmoid kernel is not valid (i.e. not the inner product of two i.e. not the inner product of two vectors) under some parameters (Vapnik, 1995).The RBF Kernel with base parameters **C=1000** and **gamma-r=0.001** was chosen as our base model. The next procedure was tested using typical split of **70%** of training data and **30%** of test data.

Other models, specifically the linear kernel and linear regression models, were chosen in order to compare the performance of the RBF kernel. LIBVSM has inbuilt libraries to support the above implementations.

3.6.4 Scaling

Scaling before applying SVM is very important. Part 2 of Sarle's Neural Networks FAQ Sarle (1997) explains the importance of this and most of considerations also apply to SVM. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. We have to use the same method to scale both training and testing data. The table and the chart below demonstrate the how the coefficient of determination changes due to change in different scaling parameters.

With the base parameters used to test the performance of data on different scales, it was observed that better results are achieved when original data is scaled by multiplying it with a factor of 100 at reasonable cost of training time.

		Coefficient of determination		
SCALE		N	N*10	N*100
TRAIN(70%)		0.859100774	0.994823645	0.997053454
TEST(30%)		0.857394917	0.994058229	0.996527775
TIME		done in 8.566s	done in 8.736s	done in 21.352s

Table 3.6.4.1 Scaling Parameters versus Accuracy

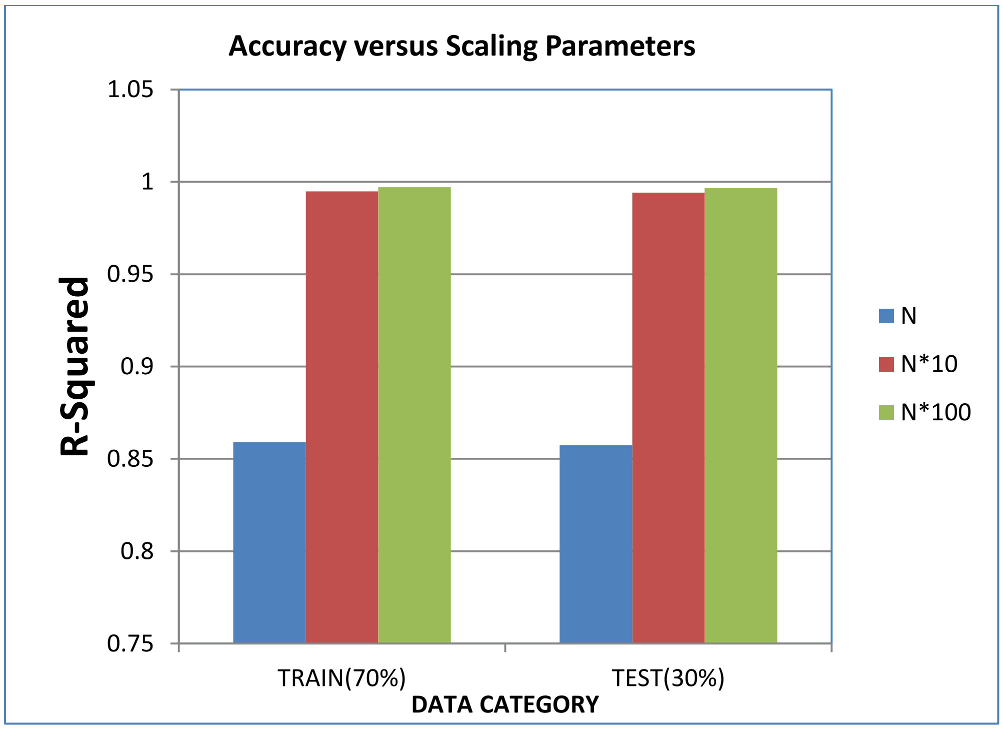


Figure 3.6.4.1 Scaling Performance

3.6.5 Cross-validation

There are two parameters for an RBF kernel: **C** and **gamma-r.** It is not known beforehand which C and gamma-r are best for a given problem; consequently some kind of model selection (parameter search) must be done. The goal is to identify good (C and gamma-r) so that the regression model can accurately predict unknown data (i.e. testing data). It may not be useful to achieve high training accuracy (i.e. a regression model which accurately predicts training data whose class labels are indeed known). As discussed above, a common strategy is to separate the data set into two parts, of which one is considered unknown. Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called **over-fitting**. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a **test set** X_test, Y_test. In order to limit problems like over-fitting, a 10 fold cross-validations was performed to the dataset of the model in the training phase (i.e. the validation dataset). This gave an insight on how the model generalized to a set of independent data set which used different parts of the input to train and test the model. The input was divided in **n** parts where **(n – 1)** parts was used to train and the last one to test the model. The training and testing processes were executed **n** times where each testing execution used a different input.

In general the performance measure reported by ***k*-fold** cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive, but does not waste too much data

3.6.6 Grid Search

Grid-search on C and gamma-r using cross-validation is recommended. Various pairs of (C and gamma-r) values are tried and the one with the best cross-validation accuracy is picked. LIBSVM provided and internally built library within python to allow one to perform an exhaustive search. The grid search was performed with the parameter indicted below.

Kernel Type	**Parameters**	**Best Parameter**
'kernel': ['rbf']}]	C: [1, 10, 100,1000,10000,100000]	C=1000
	gamma: [0.001,0.001, 0.0001]	gamma=0.0001
'kernel': ['linear']}]	C: [1, 10, 100, 1000]	C=1

Table 3.6.6.1 Grid Search to find best parameters

After performing grid search the base model was improved and the best parameters indicated on the table above were qualified for the next stage of improving accuracy by adjusting the training data size.

3.6.7 Error Adjustment

The performance of the model is measured through the use of statistical error measurements. The model error is calculated in each training and testing execution. The MSE and the RMSE is used as our performance statistics. In statistics, the mean squared error (MSE) of an estimator measures the average of the squares of the "errors", that is, the difference between the estimator and what is estimated.

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2.$$

$$\mathrm{RMSE} = \sqrt{\sum \frac{(y_{pred} - y_{ref})^2}{N}}$$

Where Y_i the observation is value and Y_i^* is the predicted value. The RMSE is the root of MSE.

Another performance index is the coefficient of determination-$\mathbf{r}^2$. This is useful because it gives the proportion of the variance (fluctuation) of one variable that is predictable from the other variable. The coefficient of determination is the ratio of the explained variation to the total variation. The coefficient of determination is such that $0< \mathbf{r}^2 <1$, and denotes the strength of the linear association between **x** and **y**. The coefficient of determination represents the percent of the data that is the closest to the line of best fit. It is a measure that allows us to determine how certain one can be in making predictions from a certain model/graph.

3.6.8 Determining the training set using learning curve method

A **learning curve** conventionally depicts improvement in performance on the vertical axis when there are changes in another parameter (on the horizontal axis), such as training set size (in machine learning). A learning curve is often useful to plot for improving performance.

For this particular research the cross validation error and the testing error was plotted versus the training size. The procedure was performed for all the chosen models and the result are were summarized as shown below.

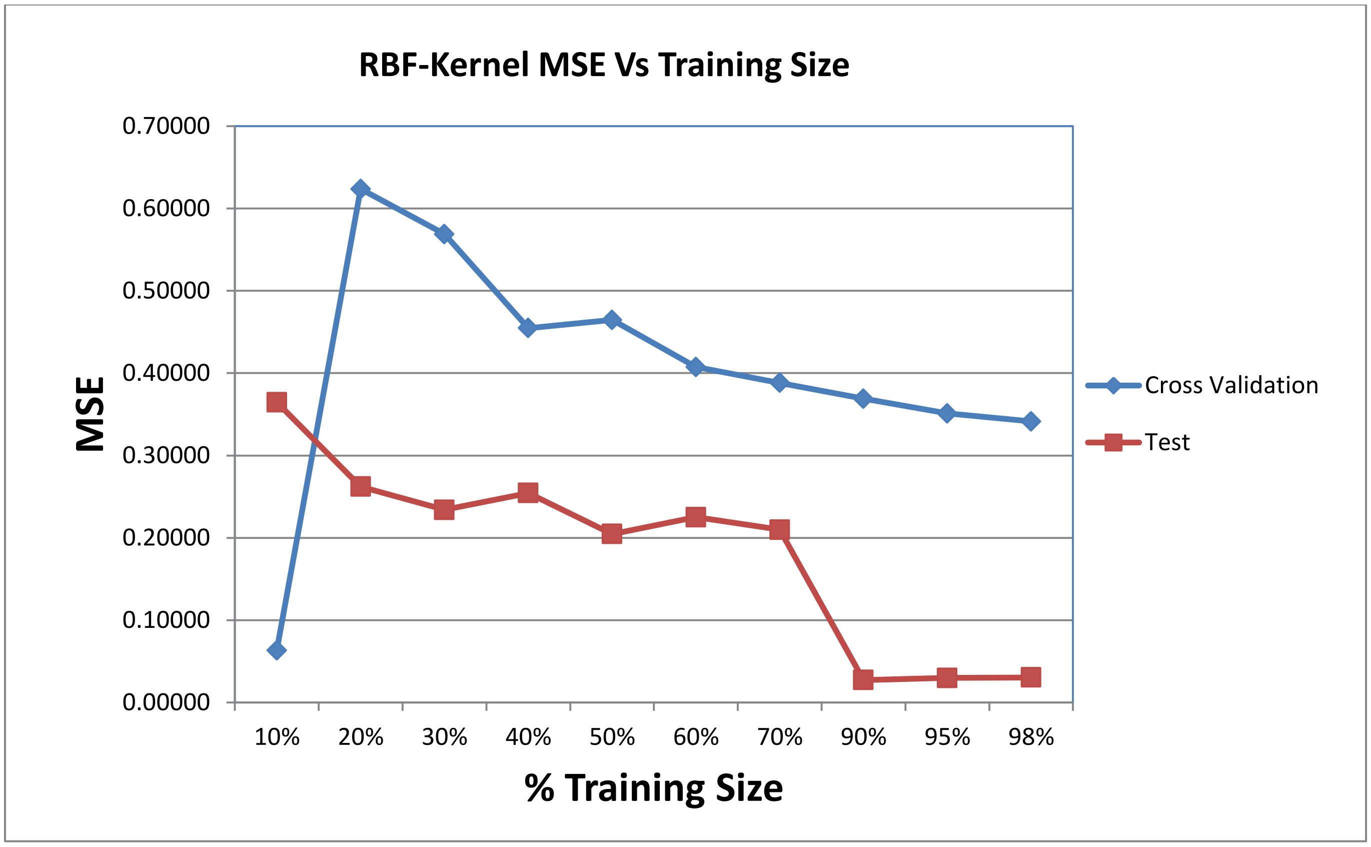


Figure 3.6.8.1 RBF Kernel Learning Curve

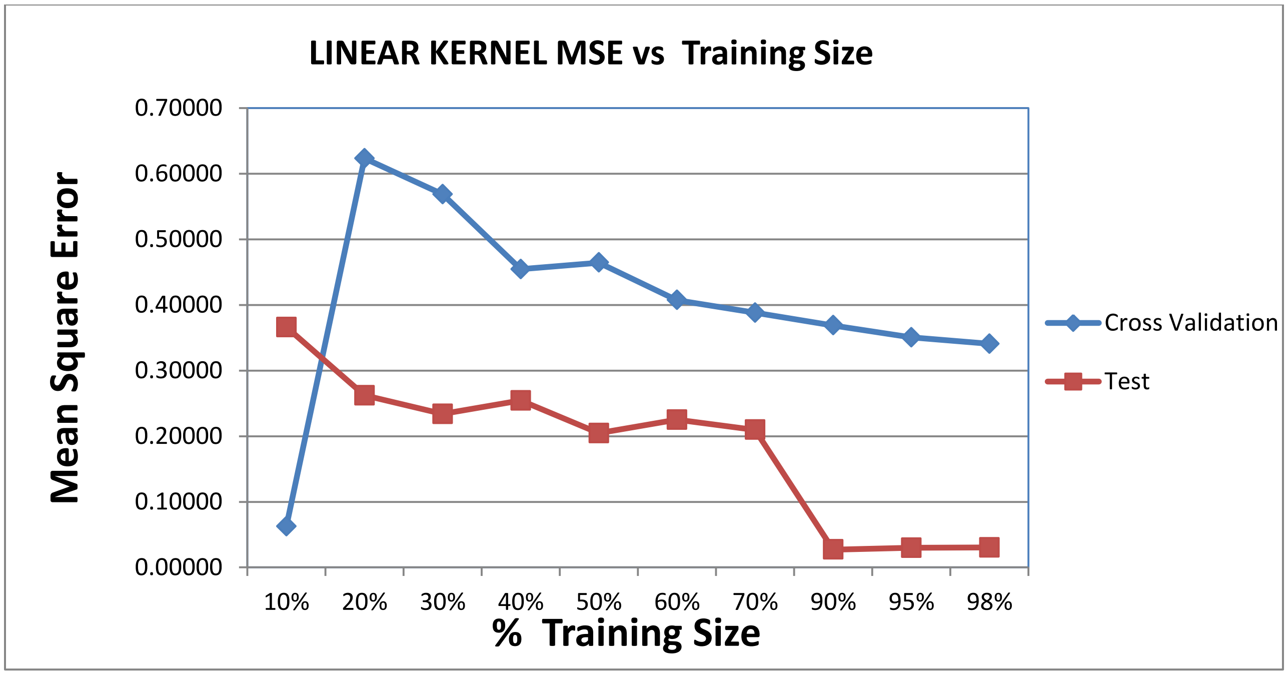


Figure 3.6.8.2: Learning curve for linear kernel

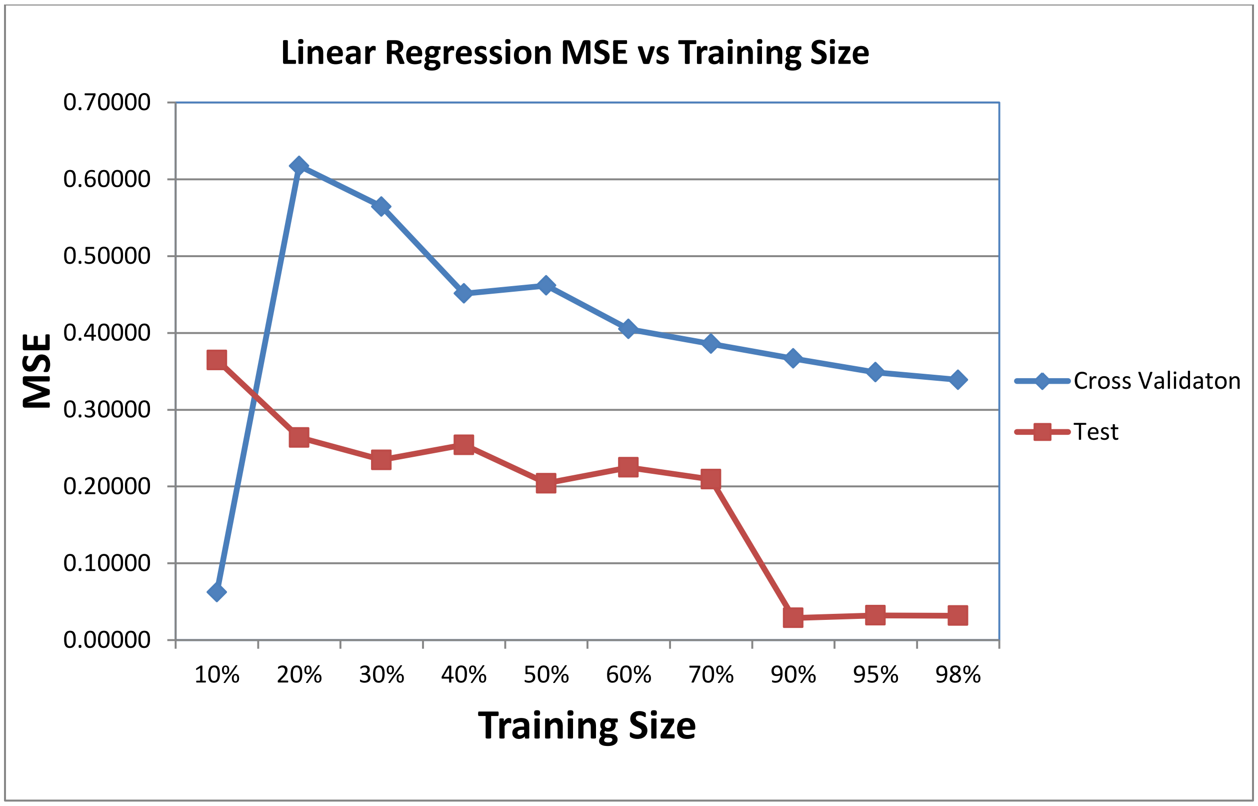


Figure 3.6.8.3: Learning curve for linear regression

From the learning curves above, it can be observed that there is a general decline on both the training and the test error. At **90%** of the training data, it is observed that the error stabilizes and this implies that increasing more training data does not significantly improve the accuracy and precision of the model. It can be then concluded that at a minimum 90% of the data can be used to train and build the prediction model combining it with the best parameters form the grid search. We also observe that the training accuracy is significantly low at **10%** of training data. We can however see that a model built at this level will not perform well on the test data and hence the parameters chosen at this level will not make a better prediction model. The same can be mentioned of the linear regression model.

Figure 3.4 and 3.5 below summarizes the both the MSE and the coefficient of determination-$\mathbf{r}^2$ results for the entire test performed at varying training sizes. See the appendix for a full table results.

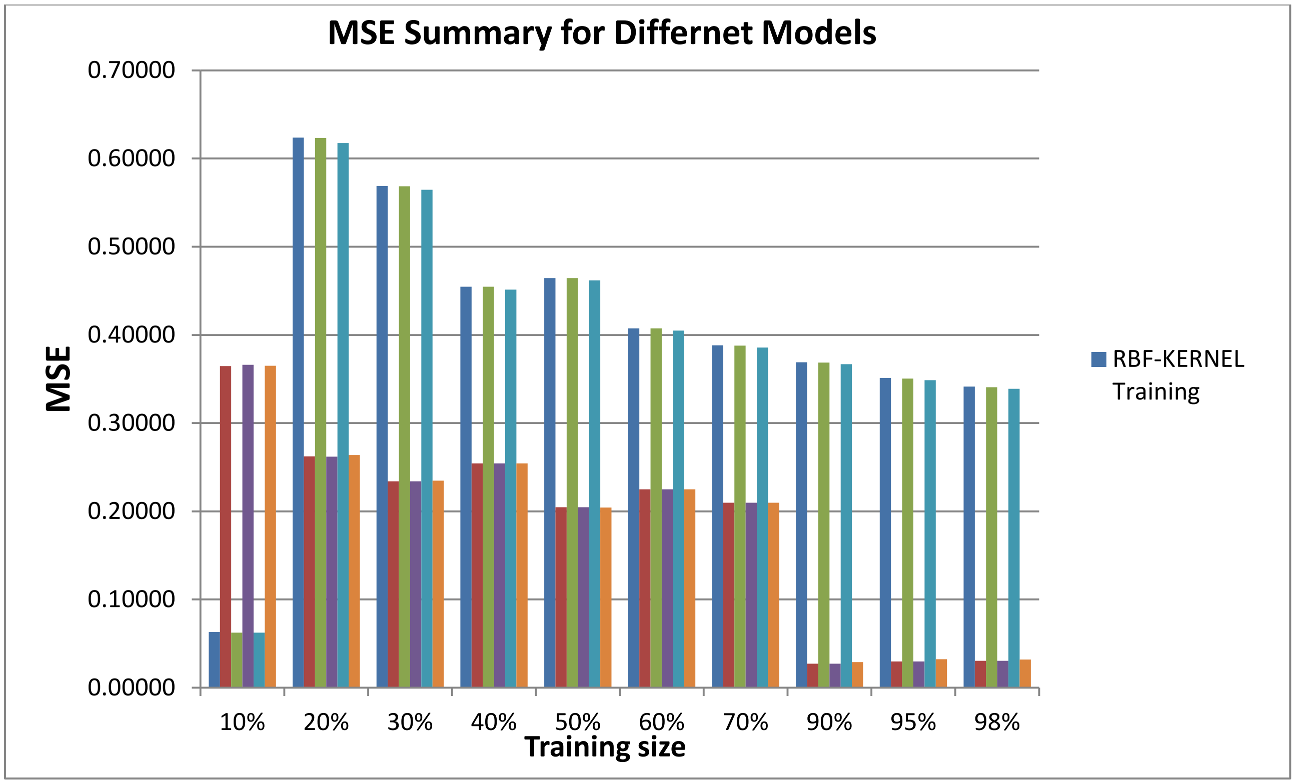


Figure 3.6.8.4 Chart summary of MSE for all the Models

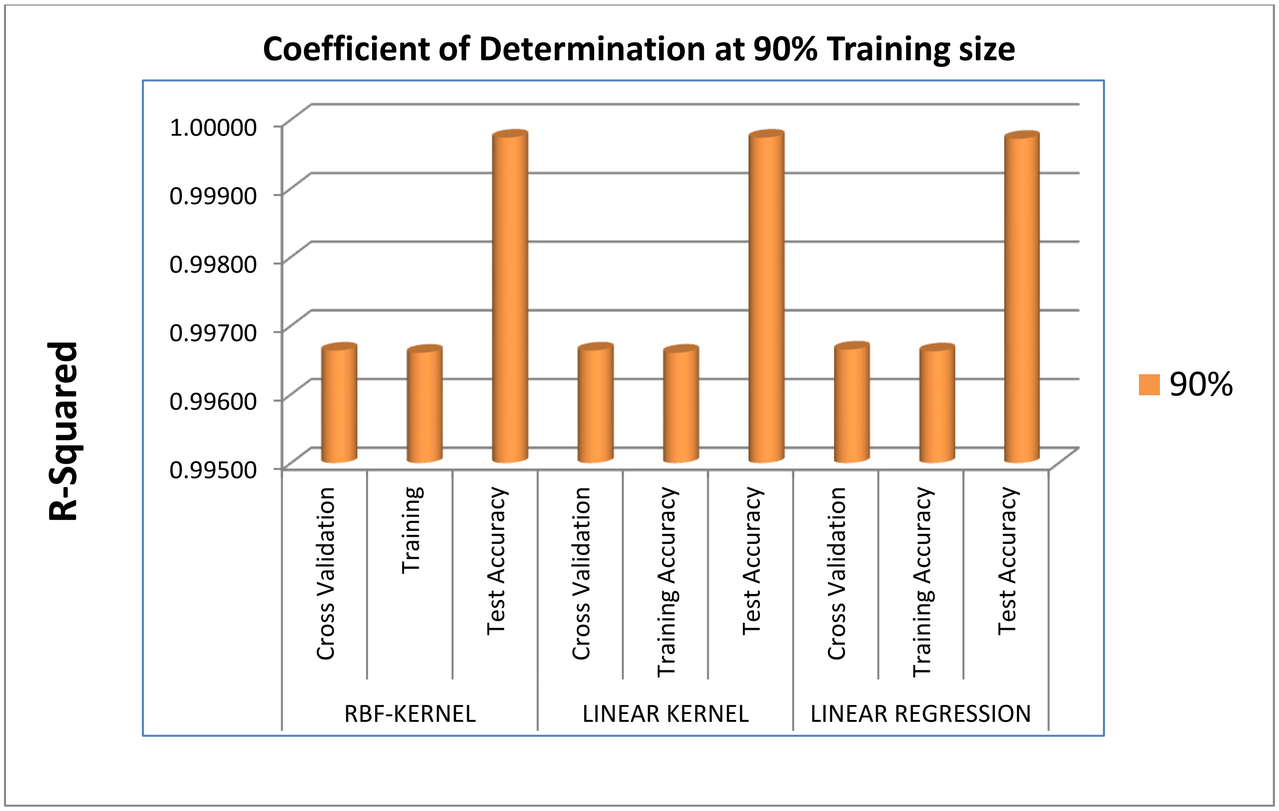


Figure 3.6.8.5: Coefficient of determination at 90% training size

3.6.9 Final Travel time prediction model parameters

After performing the above exploratory experiments, the best parameters for all the three models that can be used for training and predictions for all the sections are as summarized in the table below.

Kernel Type	Best Parameter	Training Size
Radial Basis Function-RBF	C=1000	90%
	gamma=0.0001	
Linear	C=1	90%
Linear Regression	N/A	90%

Table 3.6.9.1 Final Parameter for the Prediction Model

The model was developed and tested in LIBVSM through PYTHON programming. The prediction results were exported and the prototype developed using PHP/MySQL language.

3.7 Prototyping Methodology

This is necessary as way to demonstrate how the prediction model can be implemented in the real world of transport. A prototype of the actual system was built using the prototyping methodology.
System prototyping is a system development methodology based on building and using a model of a system for designing, implementing, testing, and installing the system. Prototyping as a methodology is important in building fast, better, more reliable, better quality systems.

Prototyping is based on building a model of a system to be developed. Moreover, the initial model should include the major program modules, the database, screens, reports, and the inputs and outputs that the system will use for communicating with other (interfacing) systems.
There are several steps that will be taken in building the Prototype model:

1. The new system requirements will be defined in as much detail as possible.
2. A preliminary design will be created for the new system.
3. A first prototype of the new system will be constructed from the preliminary design. This will be a scaled-down system, and represents an approximation of the characteristics of the final product.

3.7.1 Prototype performance

As per the prototyping methodology these were the steps used to build the prototype model:

1. The system requirements are:-
    a. Prediction of vehicle arrival time between two section of the road between Nyayo stadium and Westlands roundabout.
    b. To indicate the arrival time at each of the roundabout between the two sections above.
    c. Display google map for the above defined location
    d. Display travel time in minutes
2. A preliminary design will be created for the new system.
3. A first prototype of the new system was constructed from the above design. This is a scaled-down system, and represents an approximation of the characteristics of the final product. The interfaces are displayed below.

3.7.2 Sample Prototype Interface

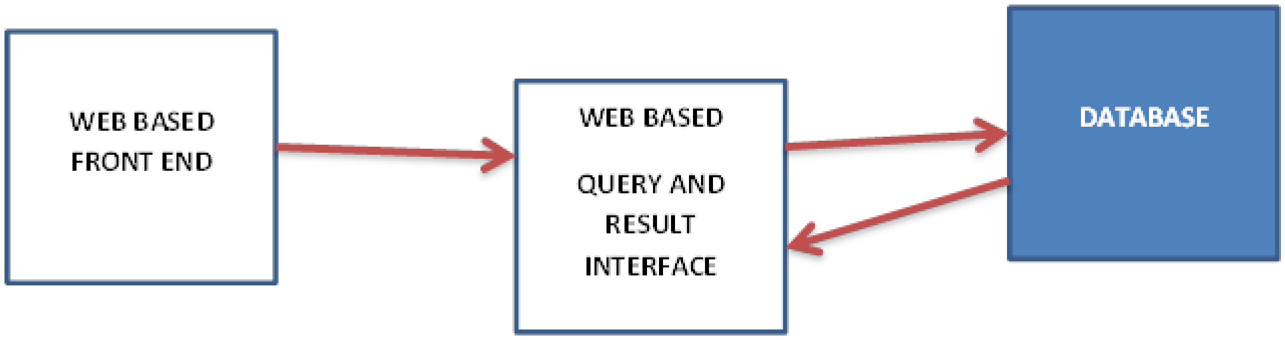
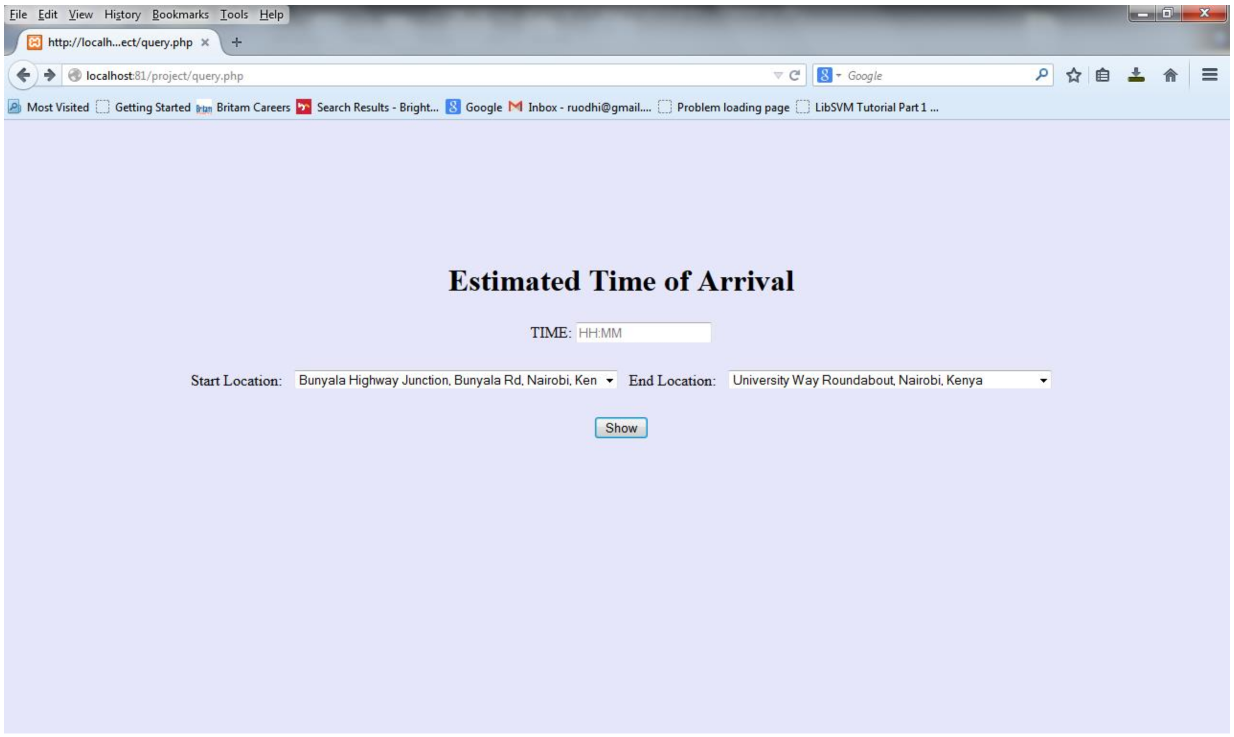


Figure 3.7.2.1: Prototype conceptual framework


File Edit View History Bookmarks Tools Help
http://localh...ect/query.php
localhost:81/project/query.php
Most Visited Getting Started Britam Careers Search Results - Bright... Google Inbox - ruodhi@gmail.... Problem loading page LibSVM Tutorial Part 1 ...

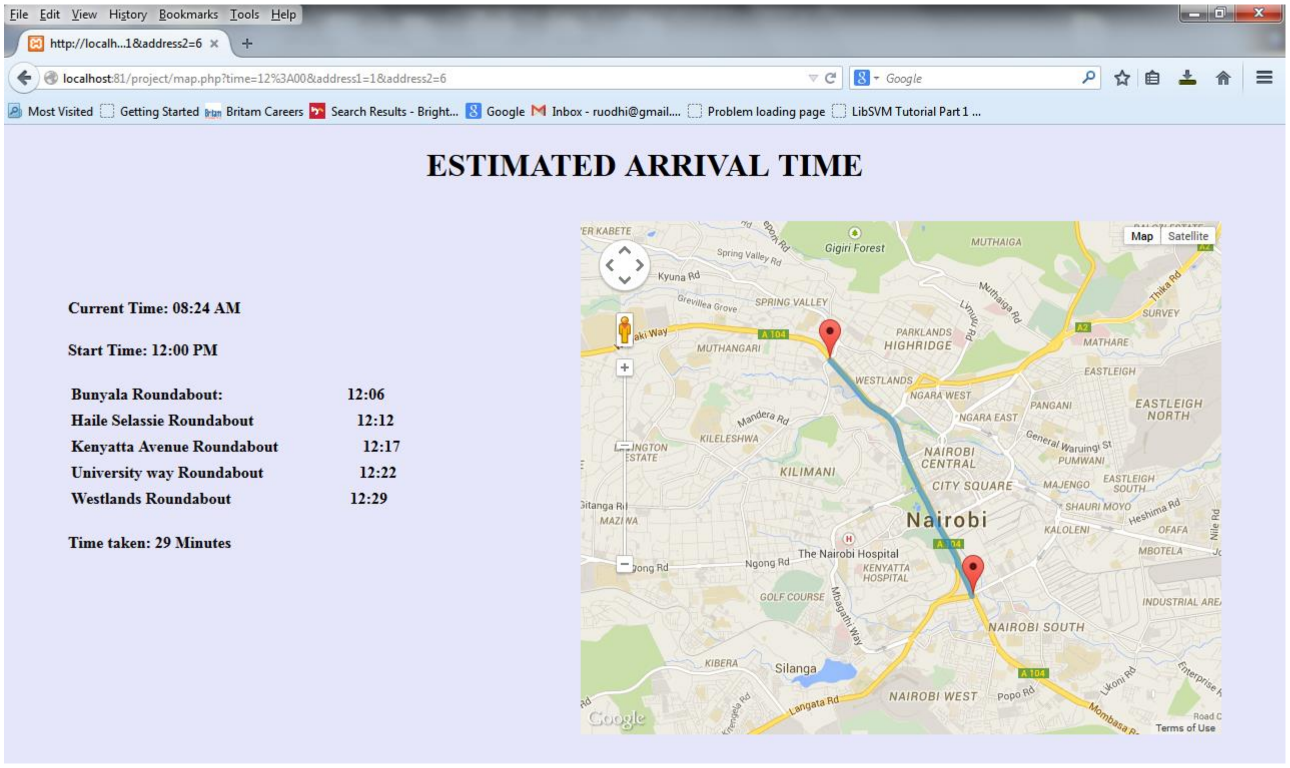
Estimated Time of Arrival

TIME: HH:MM

Start Location: Bunyala Highway Junction, Bunyala Rd, Nairobi, Ken End Location: University Way Roundabout, Nairobi, Kenya

Show

Figure 3.7.2.2 Web based front end user interface


File Edit View History Bookmarks Tools Help
http://localh...1&address2=6
localhost:81/project/map.php?time=12%3A00&address1=1&address2=6
Most Visited Getting Started Britam Careers Search Results - Bright... Google Inbox - ruodhi@gmail.... Problem loading page LibSVM Tutorial Part 1 ...
Map
Satellite
Nairobi
Westlands
Nairobi Central
City Square
Nairobi South
Kibera
Nairobi West
Google

ESTIMATED ARRIVAL TIME

Current Time: 08:24 AM

Start Time: 12:00 PM

Bunyala Roundabout:	12:06
Haile Selassie Roundabout	12:12
Kenyatta Avenue Roundabout	12:17
University way Roundabout	12:22
Westlands Roundabout	12:29

Time taken: 29 Minutes

Figure 3.7.2.3: Results interface

CHAPTER IV

MODEL EVALUATION AND RESEARCH FINDINGS

4.1 Introduction

The main purpose of the previous chapter was to build a prediction model in line with our research objectives. In this chapter the focus is to evaluate the performance of the developed model. Since data was used, a statistical measure of accuracy is defined and used. The implication of this statistical measure to our model accuracy is then discussed. This measure is then applied in comparing the accuracy of the different prediction models developed in this research study and from literature. The accuracy of the developed model on different partitions of data is later compared and discussed. The research results are then analyzed and the research findings presented. The results obtained in this study are then compared to other results obtained from literature. The findings will seek to answer the research questions that guided the study.

4.2 Model Performance

It is important to evaluate the developed prediction models in terms of prediction accuracy. The SVR model has been evaluated for its performance and compared with that of linear regression travel time model. The accuracy of the developed model is further compared with the values obtained from literature.

Root Mean Square Error (RMSE) is one of the most widely used statistical measures. RMSE measures how much error there is between two sets of paired data. RMSE compares a predicted value and an observed value. In this research we compare the predicted value to the observed value. Root mean square error takes the difference for each predicted value and observed value. This is how RMSE is calculated. Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and is the most important criterion for fit if the main purpose of the model is prediction (Karen, 2015).

RMSE Formula:

$$\text{RMSE} = \sqrt{\sum \frac{(y_{pred} - y_{ref})^2}{N}}$$

Where,

Y_{pred} is the predicted value

Y_{ref} is the observed value

N is the number of datasets

From the table below the summary of the RMSE for the three prediction models from the tests performed by the chosen final parameters across all the five sections can be compared. On average the RBF and Linear kernel performed within the acceptable accuracy as compared to the linear regression across the five sections. The table of results is summarized in the Appendix section for further details.

	TEST ERROR	PERCENTAGE RMSE		
	MODEL	RBF-KERNEL	LINEAR KERNEL	LINEAR REGRESSION
Section One	Nyayo-Bunyala	16.50%	16.51%	17.01%
Section Two	Bunyala-Haile	20.17%	20.47%	20.27%
Section Three	Haile-Kenyatta Ave	14.68%	14.60%	14.60%
Section Four	Kenyatta Ave-University Way	15.15%	15.42%	15.38%
Section Five	University Way-Westlands	26.41%	26.12%	N/A

Table 4.2.1: Percentage RMSE

It was generally found that prediction of travel time in the study area across all the five sections could be given with an RMSE value of **18.01 %** using the SVR model. This average is as a result of two sections yielding slightly higher values than the average as compared to the other sections that posted lower values. It was discovered that the linear SVR model gave a better prediction errors of travel time.

The coefficient of determination is a measure of how well the regression line represents the data. If the regression line passes exactly through every point on the scatter plot, it would be able to explain all of the variation. The further the line is away from the points, the less it is able to explain. The coefficient of determination is such that $0 < \mathbf{r}^{2} < 1$, and denotes the strength of the linear association between two variables. It represents the percent of the data that is the closest to the line of best fit.

From the table below, it was found that prediction of travel time in the study area across all the five sections yielded an average coefficient of determination of **99.1%** for the entire predictive model. This indicates a good predictive model in general.

This measure indicates that **99.1%** of the total variation in the predicted values can be explained by the relationship between the observed and predicted values. Only **0.9%** of the total variation in the predicted values remains unexplained. This generally indicates a good fit for all our predictive models across all the five sections.

		Coefficient of Determination-R Squared		
Data Train 90%	**Description**	**RBF-KERNEL**	**LINEAR KERNEL**	**LINEAR REGRESSION**
		Test Accuracy	Test Accuracy	Test Accuracy
Section One	**Nyayo-Bunyala**	99.974%	99.974%	99.972%
Section Two	**Bunyala-Haile**	99.961%	99.960%	99.961%
Section Three	**Haile-Kenyatta Ave**	99.979%	99.980%	99.979%
Section Four	**Kenyatta Ave-University Way**	99.762%	99.763%	99.764%
Section Five	**University Way-Westlands**	94.165%	94.177%	N/A

Table 4.2.2 Coefficient of Determination

4.3 Model Results for several Sections of the Road

The prediction algorithm developed was subjected to the **10%** test data that was not used in training to be able to predict the arrival time of trucks. The summary tables below display all the results that were achieved form the test results.

According to the results presented it can be concluded that on average all the three models can predict the arrival time for section one (Nyayo round about to Bunyala roundabout) within one or two minutes on average of the actual arrival time. This is also true for all the other four sections with exception of the linear regression model that performed poor for section five. This is indicative of a good predictive model considering that the average RMSE obtained was **18.01%** with $\mathbf{r^2}$ of **99%**

The poor performance depicted by the linear regression model for this section indicates that the traffic patterns along that section cannot be described using a linear model. It can therefore be concluded that the arrival time for this section cannot be linearly predicted or explained.

No	Nyayo	Bunyala	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	3:29:00 PM	3:37:00 PM	3:35:36 PM	3:35:32 PM	3:36:23 PM
2	3:11:00 PM	3:16:00 PM	3:17:34 PM	3:17:31 PM	3:18:21 PM
3	1:10:00 PM	1:15:00 PM	1:16:22 PM	1:16:26 PM	1:17:07 PM
4	1:19:00 PM	1:24:00 PM	1:25:23 PM	1:25:27 PM	1:26:08 PM
5	2:56:00 PM	3:01:00 PM	3:02:32 PM	3:02:30 PM	3:03:20 PM
6	2:06:00 PM	2:10:00 PM	2:12:26 PM	2:12:28 PM	2:13:14 PM
7	1:39:00 PM	1:45:00 PM	1:45:24 PM	1:45:27 PM	1:46:10 PM
8	10:56:00 AM	11:00:00 AM	11:02:24 AM	11:02:21 AM	11:02:51 AM
9	11:01:00 AM	11:10:00 AM	11:07:24 AM	11:07:21 AM	11:07:52 AM
10	1:52:00 PM	2:00:00 PM	1:58:25 PM	1:58:28 PM	1:59:12 PM

Table 4.3.1: Prediction Results section one (Nyayo-Bunyala roundabout)

No	Bunyala	Haile Selassie	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	3:37:00 PM	3:46:00 PM	3:44:04 PM	3:43:36 PM	3:44:14 PM
2	3:16:00 PM	3:21:00 PM	3:22:53 PM	3:22:35 PM	3:23:11 PM
3	1:15:00 PM	1:25:00 PM	1:21:20 PM	1:21:31 PM	1:21:57 PM
4	1:24:00 PM	1:31:00 PM	1:30:21 PM	1:30:31 PM	1:30:58 PM
5	3:01:00 PM	3:06:00 PM	3:07:46 PM	3:07:35 PM	3:08:09 PM
6	2:10:00 PM	2:13:00 PM	2:16:29 PM	2:16:33 PM	2:17:03 PM
7	1:45:00 PM	1:56:00 PM	1:51:24 PM	1:51:32 PM	1:52:00 PM
8	11:00:00 AM	11:07:00 AM	11:06:25 AM	11:06:26 AM	11:06:40 AM
9	11:10:00 AM	11:18:00 AM	11:16:24 AM	11:16:26 AM	11:16:42 AM
10	2:00:00 PM	2:07:00 PM	2:06:27 PM	2:06:33 PM	2:07:02 PM

Table 4.3.2: Prediction Results section two (Bunyala-Haile Selassie roundabout)

No	Haile Selassie	Kenyatta Avenue	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	3:46:00 PM	3:54:00 PM	3:53:03 PM	3:52:44 PM	3:53:02 PM
2	3:21:00 PM	3:27:00 PM	3:27:49 PM	3:27:40 PM	3:27:57 PM
3	1:25:00 PM	1:30:00 PM	1:31:02 PM	1:31:20 PM	1:31:34 PM
4	1:31:00 PM	1:39:00 PM	1:37:04 PM	1:37:21 PM	1:37:35 PM
5	3:06:00 PM	3:11:00 PM	3:12:42 PM	3:12:37 PM	3:12:54 PM
6	2:13:00 PM	2:20:00 PM	2:19:18 PM	2:19:28 PM	2:19:43 PM
7	1:56:00 PM	2:01:00 PM	2:02:12 PM	2:02:25 PM	2:02:40 PM
8	11:07:00 AM	11:09:00 AM	11:12:38 AM	11:12:55 AM	11:13:07 AM
9	11:18:00 AM	11:25:00 AM	11:23:39 AM	11:23:57 AM	11:24:09 AM
10	2:07:00 PM	2:16:00 PM	2:13:16 PM	2:13:27 PM	2:13:42 PM

Table 4.3.3: Prediction Results section three (Haile Selassie-Kenyatta roundabout)

No	Kenyatta Avenue	University Way	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	3:54:00 PM	4:00:00 PM	4:00:42 PM	4:00:27 PM	4:00:45 PM
2	3:27:00 PM	3:33:00 PM	3:33:23 PM	3:33:22 PM	3:33:40 PM
3	1:30:00 PM	1:36:00 PM	1:35:35 PM	1:36:02 PM	1:36:16 PM
4	1:39:00 PM	1:46:00 PM	1:44:37 PM	1:45:04 PM	1:45:18 PM
5	3:11:00 PM	3:20:00 PM	3:17:13 PM	3:17:20 PM	3:17:36 PM
6	2:20:00 PM	2:25:00 PM	2:25:49 PM	2:26:11 PM	2:26:26 PM
7	2:01:00 PM	2:07:00 PM	2:06:43 PM	2:07:08 PM	2:07:22 PM
8	11:09:00 AM	11:18:00 AM	11:14:27 AM	11:14:38 AM	11:14:47 AM
9	11:25:00 AM	11:28:00 AM	11:30:27 AM	11:30:40 AM	11:30:50 AM
10	2:16:00 PM	2:25:00 PM	2:21:48 PM	2:22:10 PM	2:22:25 PM

Table 4.3.4: Prediction Results section four (Kenyatta avenue-University way)

No	University Way	Westlands	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	4:00:00 PM	4:11:00 PM	4:08:56 PM	4:08:40 PM	9:09:32 PM
2	3:33:00 PM	3:40:00 PM	3:41:33 PM	3:41:36 PM	9:13:03 PM
3	1:36:00 PM	1:50:00 PM	1:43:36 PM	1:44:15 PM	9:28:16 PM
4	1:46:00 PM	1:53:00 PM	1:53:38 PM	1:54:16 PM	9:26:58 PM
5	3:20:00 PM	3:27:00 PM	3:28:23 PM	3:28:33 PM	9:14:44 PM
6	2:25:00 PM	2:33:00 PM	2:32:51 PM	2:33:23 PM	9:21:53 PM
7	2:07:00 PM	2:16:00 PM	2:14:44 PM	2:15:20 PM	9:24:14 PM
8	11:18:00 AM	11:25:00 AM	11:25:33 AM	11:25:50 AM	9:46:13 PM
9	11:28:00 AM	11:34:00 AM	11:35:32 AM	11:35:52 AM	9:44:55 PM
10	2:25:00 PM	2:35:00 PM	2:32:51 PM	2:33:23 PM	9:21:53 PM

Table 4.3.5: Prediction Results section five (University way-Westlands)

From the results obtained, it can be observed that all the models performs generally well and this can attributed to following best practices when handling SVM related algorithms. Chih discusses that researchers who are not familiar with SVM often get unsatisfactory results since they miss some significant steps. This research adopted the guidelines offered and the results achieved produced models with better accuracy and results (Chih, Chang, & Lin, 2010).

Predicted values for all the models were further compared for different sections, the sections have already been defined in the third chapter of this report, covering the entire study area. For all the sections, ten (10) predictions results were presented from the test data. Since the algorithm used split the data according to its internally built methods, the times displayed are not representative of the entire day but part of it. More predicted values are available in the appendix section. The table below indicated the actual time the truck entered and exited the study section. The exit time is then compared to the predicted time for each of the three models developed in this study.

4.4 Performance comparison between partitioned and full data

Data was partitioned into three parts based on the general observed traffic patterns along the route of study. The three partitions were informed by the fact that traffic generally is heavier in the morning and in the evening when the highway is mostly used by motorists heading to and from work. The other portion captures the time period when traffic is either normal or better compared to the morning and evening traffic. Coefficient of determination was used as a measure of performance between the two data sets under comparison and the results were summarized below.

See Appendix B for results for each partitioned data.

		Average R^2 for partitioned data			
Data Train 90%	**Description**	**RBF-KERNEL**		**LINEAR KERNEL**	
		Cross Validation Accuracy	Test Accuracy	Cross Validation Accuracy	Test Accuracy
Section One	**Nyayo-Bunyala**	0.9867	0.9939	0.9867	0.9939
Section Five	**University Way-Westlands**	0.4420	0.5013	0.5855	0.3576

Table 4.4.1: Average results for specified sections for the partitioned data

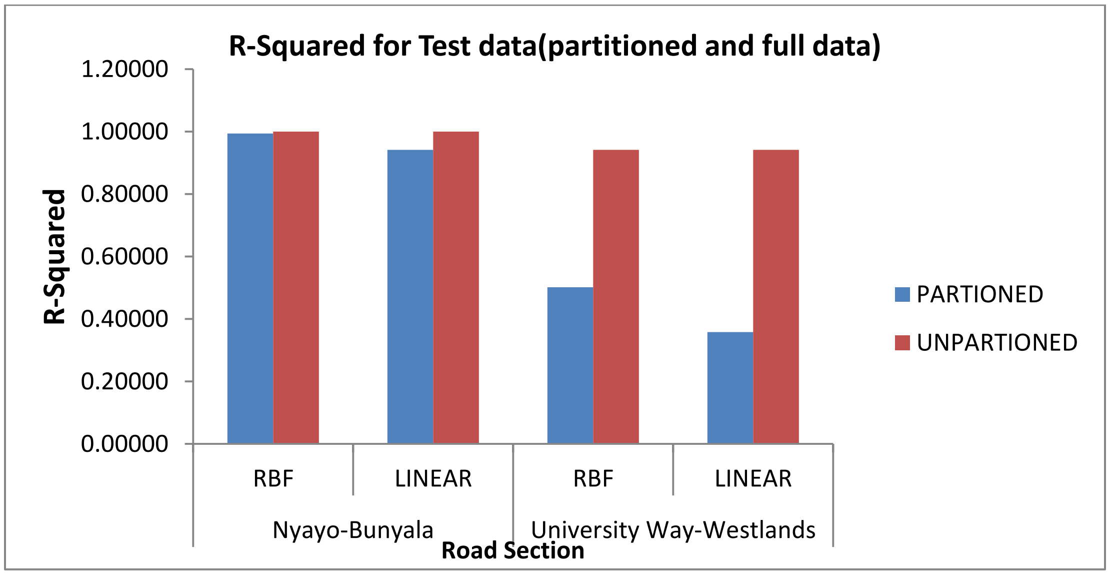


Figure 4.4.1: R-squared for test data for portioned and full data

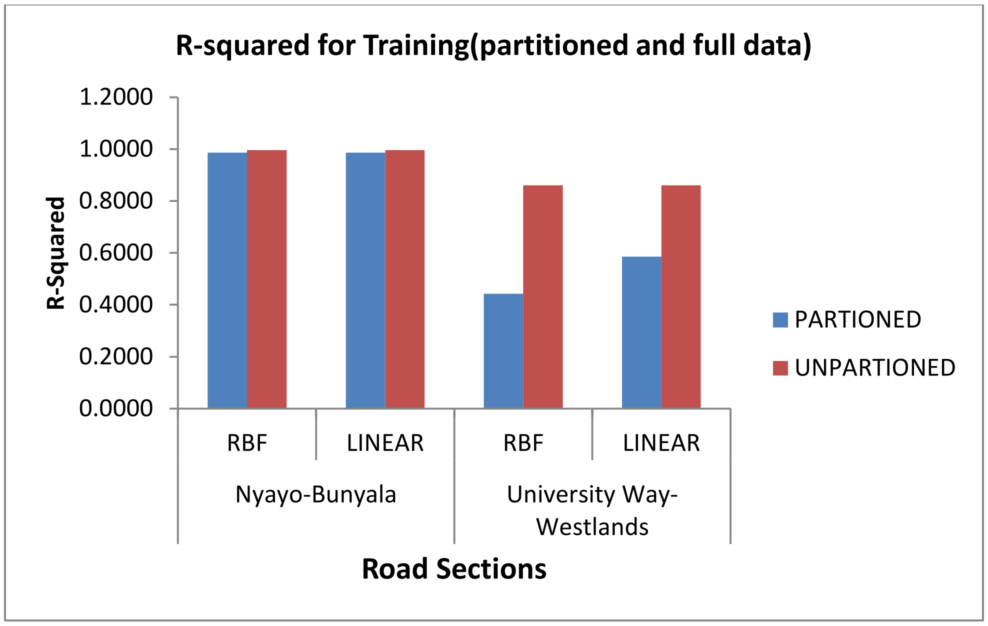


Figure 4.4.2: R-squared for training (full and partitioned data)

When these results were compared to the results obtained when the analysis was done with full dataset it can be noted that training based on full dataset produced a model that performed much better both in training accuracy and test accuracy. This can be either attributed to having reduced data for both training and testing that can have an effect on accuracy. In such a case collecting more data can be advised in order to improve the accuracy prediction. The other conclusion can be that clustering of data in view of traffic pattern does not necessarily imply that we can yield an improved model.

4.5 Comparison between the SVR Model and previous model

Zegeye and Null researched on the use of dynamic artificial neural network travel time prediction model for buses using only GPS data in Brazil. The goal of his research was to develop a dynamic model that can provide accurate prediction for the ETA of a bus at a given bus stop using global positioning system (GPS) data which is collected in Brazil. He designed an artificial neural network (ANN) based prediction model because of its ability to solve complex non-linear relationships. Usually a low MAPE value is desirable for an algorithm. The research study posted an average MAPE result of 18.3% (Zegeye & Nall, 2014). He further developed an historical model as a comparative model to the key ANN model and concluded that in general, the ANN outperformed the average approach in terms of both prediction accuracy and robustness. When the MAPE for the developed SVR models in this research were calculated, the average value was found to be less than 1%, indicating that the SVR model would perform better than the ANN model. This research though still recommends a comparative study between the two models when similar datasets are used.

In 2003, Wei and co researched on the use of SVR for travel time prediction for long distance travel along a freeway and averaged 13.91% in RMSE (Wei, Wu, Ming-Hua , Chang, & Ho, 2003). The research developed two other baseline predictors to compare the performance of the SVR model. This research posted RMSE of 18.03%. Compared to other baseline predictors, the results showed that the SVR predictor can reduce significantly the root mean squared errors of predicted travel times.

4.6 Summary

In this chapter, the three prediction models developed in chapter III were evaluated and statistically tested. The SVR models gave better results than the linear regression model across all the five sections in term of prediction accuracy. It is found that RMSE increases as prediction time increases or when there are a lot of traffic dynamics along a section. It was also seen that clustering data leads to larger coefficient of determination as compared to ungrouped data and thus larger errors.

It was found that the MAPE of SVR gives better predictions results that the previously researched models like ANN and historical based models. As a result, the SVR method was adopted as the preferred prediction method. In chapter V, three issues related to truck arrival time will be discussed under conclusions and recommendations: the adoption of the prediction model, the feasibility for real-time application and the opportunity for further research.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

5.1 Introduction

Accelerating traffic growth over the past decades has been a threat to the quality of life of people in many countries. Traffic congestion leads to a decrease in accessibility, travel time loss and air pollution. In developed countries most people still use private vehicles while in developing countries the level of vehicle ownership is rising at a faster rate.
Government approximations has revealed that traffic jams cost 50 million shillings ($578,000) a day in lost productivity in the city (Welsh 2011). Unpredictable and growing traffic jam is a drawback to the quality of life of people in many urban centers, including the city of Nairobi, over the past few decades.

While the government is still struggling with ways and means to reduce congestions in major urban areas especially in the capital city-Nairobi, there is need for a consumer based solution that would help drivers make informed decisions while driving in and around the city of Nairobi. This was the motivation behind this research problem.

This research set out to use real time data to develop and test time prediction model that provides an estimated arrival time for trucks between two clearly defined set of locations depending on traffic conditions. This involved collecting real time data to design and test different predictive algorithms. Support vector regression (SVR) was used to build different predictive models to predict the travel time of a truck on a highway then a web based prototype system was built to demonstrate how the model can be adopted in implementing a real world system. It is part of the background work required to implement an actual travel time predictions system.

This research explored mainly the use of different SVR algorithms to develop travel time prediction models for trucks under disciplined traffic conditions given real time data. This was achieved through the analysis of real time data collected along road sections of Uhuru highway, Nairobi on specific days and time of the week for a period of two weeks. The analysis of the data involved using the aggregated data to train an SVR Model that was later adopted for use as a time predictions model. Through the use grid-search, cross-validation and learning curve the right predictive model with better performance accuracy was chosen and adopted.

The predicted travel time of the route under consideration were compared with the collected data. The performance of the model was also tested and compared with linear regression model and results from literature, where the predicted travel time takes the shape of a linear function between two defined points under consideration. Prediction accuracies, coefficient of determination and RMSE, were used as performance measure.

5.2 Conclusions

The developed SVR time predictions model demonstrates much success in estimating truck arrival time for the sections under study and performs better than the previously developed models for travel time predictions. Zegeye and Null researched on the use of dynamic artificial neural network travel time prediction model for buses using only GPS data in Brazil and got an average MAPE result of **18.3%** (Zegeye & Nall, 2014). With an average RMSE value of **18.01%** and a MAPE of less than **1%** across the five sections under this case study justifies the need to adopt the use of support vector regression (SVR) to build the travel time predictive model. Previous research that adopted the use of SVR model further strengthens this conclusion. In 2003, Wei and co researched on the use of SVR for travel time prediction for long distance travel along a freeway and averaged **13.91%** in RMSE (Wei, Wu, Ming-Hua , Chang, & Ho, 2003). This is the main result achieved in this research project.

The exploration of three different predictive algorithm i.e. RBF, Linear SVR and Linear regression provides the research with alternatives models that can be adopted for each sections. On average it is clear the Linear kernel algorithm (SVR) preformed much better compared to the other two models across the five sections. This can be justified by the fact that its average predictive accuracy value of **r^2=99.98%** was way above the others.

The web based application used as prototype to this project makes a strong case that this project can be presented as a commercial venture. From the prototype we clearly see that the truck arrival time varies depending on the time of day and the distance of travel. The addition of arrival time for intermediate sections helps improve decision making on which sections to avoid or use. It is considered that the main objective of this project as indicated above was technically accomplished.

5.3 Significance of this result in general and implications of the findings

The results obtained from the overall study are promising and build a strong case for the use of SVR based framework to be adopted for developing travel time prediction systems. The model can be used to implement real time web-based or mobile application systems to predict the arrival time of vehicles between points in the study area where there is disciplined traffic flow. The implementation of this system can improve reliability of the transport system, thus helping drivers make better decisions. With an appropriate data collection scheme, it can be claimed that the results obtained can be further improved and the model adopted for commercial consumption

Certain limitations were identified in the study: Research period, funding, data collection and analysis.

1. Funding

This combined with a limited period to conduct the research contributed to scoping the project within feasible limit. This research project was funds intensive especially due to the unavailability of secondary data for the nature of this project and thus we used human resource to collect primary data which was a cheaper and more convenient option to collect primary data.

2. Data collection tools

Observation as a data collection method required the use of human objects to collect raw data. While this method has several advantages as outlined in the methodology section, it carries along with it the errors introduced by human objects that results from fatigue among other factors. However, the possibility of bias was minimized by the use of strategies such as trustworthiness and intuition throughout the study. This method combined with funding limitations scoped the research to be narrowed to trucks only as described in the methodology chapter.

Further to this, data could only be analyzed within the conditions (weather and human) presented to the research project during the data collection period. The availability of limited tools to transform time series data in a format understandable by SVM restricted the research to the use of spreadsheets and that carries along with it the limitations.

5.4 Recommendations for future work

In the light of the limitations identified and the findings of the study. The researcher makes the following recommendations for use of support vector regression to design a model for travel time predictions.

1. Due to complex nature of the traffic data, a single model or algorithm cannot achieve robust and feasible results for all roads, traffic conditions and patterns. There is an increasing trend to utilize hybrid algorithms to improve the prediction accuracy (Altinkaya et al 2013). This research thus recommends exploring different methods both individually and as a hybrid as route for future research. We further recommend studying other factors that contribute to travel time variation and adopt them as variables in building predictive models. This can improve the model further and make it more relevant when predicting time during events and changing seasons.
2. Developing a mobile application as an implementation of the developed travel time prediction model can push this research towards a commercial venture. This is supported further by the explosive demand for mobile devices in Kenya. For this to be realized the recommendations stated above must first be implemented
3. Adopting the use of technology to ease and improve data collection. Harnessing and storing data from the digital speed governors can be a good source. This will help to improve the accuracy of data collected and also reduce both the budget and time required to collect more data for use in machine learning methods. The large dataset obtained from this secondary data can them be partitioned as defined by several predetermined conditions and analyzed to help build a more robust predictive model. This will thus eliminate the challenge of funding even if the magnitude of data increases exponentially.

5.5 Summary

In this chapter the conclusions derived from the findings of this study on the use of support vector regression-a supervised machine learning approach algorithm to predict vehicle arrival time are outlined. The conclusions were based on the purpose, objectives and results of the study. The implications of these findings and the resultant recommendations were also explained. Recommendations were based on the conclusions and purpose of the study.

LIST OF SOURCES

Sayad, S. D. (2010). *An Introduction to Data Mining*. Retrieved April 4th, 2015, from http://www.saedsayad.com/: http://www.saedsayad.com/support_vector_machine_reg.htm

Al-Deek, H, M, D., & M, W. (1998). Travel Time Prediction with Non-Linear Time Series. . *5th International Conference on Applications of Advanced Technologies in Transportation* (pp. pp.317-324.). Newport Beach, CA: Proceedings of the ASCE.

Bandla. (2014, November 28th). *Microsoft Bing Maps can now predict road traffic condition across the globe.* Retrieved April 24th, 2015, from http://www.gadgetdetail.com: http://www.gadgetdetail.com/microsoft-bing-maps-can-now-predict-road-traffic-condition-across-globe

Barth, D. (2009, August 25th). *The bright side of sitting in traffic: Crowdsourcing road congestion data.* Retrieved April 4th, 2015, from http://googleblog.blogspot.ca: http://googleblog.blogspot.ca/2009/08/bright-side-of-sitting-in-traffic.html

Chickering, D. M., Heckerman, D., & Meek, C. (1997). A Bayesian approach to learning Bayesian networks . *Proceedings of UAI 97*, (pp. 80-89).

Chien, S.I.J., Ding, Y., Wei, & C. (2002). Dynamic Bus Arrival Time Prediction with Artificial Neural Networks. *Journal of Transportation Engineering, Volume 128, Number 5*, pp. 429-438.

Chih, W., Chang, C.-C., & Lin, C.-J. (2010). *A Practical Guide to Support Vector Classi.* Taiwan: National Taiwan University.

Elyse, B. (2013, December 27th). *Want to know how Google Maps calculates ETA? Ex-Googler reveals all.* Retrieved July 7th, 2014, from http://www.pocket-lint.com: http://www.pocket-lint.com/news/126071-want-to-know-how-google-maps-calculates-eta-ex-googler-reveals-all

Horvitz , E., Apacible , J., Sarin, R., & Liao, L. (2002). *Prediction, Expectation, and Surprise: Methods, Designs, and Study of a Deployed Traffic Forecasting Service.* Redmond, Washington: Microsoft.

Jeong , R. H. (2004). *The Prediction of Bus Arrival time Using Automatic Vehicle Location Systems Data.* Texas: A Ph.D. Dissertation at Texas A&M University.

Karen. (2015). *Assessing the Fit of Regression Models*. Retrieved 6 1, 2015, from The Analysis Factor: http://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/

Karthik, S., & Srikanth, R. (2014, January 21). *Now, apps for live traffic feed.* Retrieved April 4th, 2015, from http://www.thehindu.com: http://www.thehindu.com/news/cities/chennai/now-apps-for-live-traffic-feed/article5598198.ece

Ramakrishna, Y., P. Ramakrishna, V. Lakshmanan, & R. Sivanandan. (2006). Bus travel time prediction using GPS Data. . *Proceedings, Map India (2006).* .

Ruimin, L. (2004). *Examining travel time variability using AVI data.* Melbourne: INSTITUTE OF TRANSPORT STUDIES.

Shalaby, A., & Farhan, A. (2004). Bus Travel Time Prediction for Dynamic Operations Control and Passenger Information Systems. *82nd Annual Meeting of the Transportation Research Board.* Washington D.C: National Research Council.

Smith, B. L., & Demetsky, M. J. (1995). Short-Term Traffic Flow Prediction: Neural Network Approach. *Journal of Transportation Research Board*, pp.98-104.

Thomas, T., Weijermars, W. A., & Van Berk, C. E. (2010). Predictions of urban volumes in single time series. *IEEE Transactions on Intelligent Transportation Systems*, 11(1): 71-80. .

Tu, H. (2008). *Monitoring Travel Time Reliability on Freeways.* Delft: Delft University of Technology.

Van Zuylen, H. (2004). *The Effect of Irregulations of Travel Times on Departure Time Choice.* Christchurch, New Zealand: 2nd International Symposium on Transportation Network Reliability (INSTR), Edited by Nicholson and Dantas. .

Wei, C.-C., Wu, C.-H., Ming-Hua , C., Chang, M.-H., & Ho, J.-M. (2003). *Travel Time Prediction with Support Vector Regression.* Taiwan: Institute of Information Science.

Welsh, J. (2011, September 8th). *IBM Commuter 'Pain Index' Shows Worst Cities for driving.* Retrieved July 2nd, 2014, from http://blogs.wsj.com: http://blogs.wsj.com/drivers-seat/2011/09/08/ibm-commuter-pain-index-shows-worst-cities-for-driving/

Williams, B., & Hoel, L. (2003). Modeling and forescating vehicle traffic flow as a seasonal arima process: Theoretical basis and empirical results. . *Journal of Transportation Engineering*, 129(6):664–672.

Zegeye, K. G., & Nall, T. (2014). Artificial Neural Network Travel Time Prediction Model for Buses Using Only GPS Data. *Journal of Public Transportation*, Vol. 17, No. 2.

APPENDICES

APPENDIX A: TRAINING DATA AND ANALYSIS RESULTS

1. SUMMARIZED AGGREGATED DATA (50 ENTRIES)

No.	Date	Day	Plate	Nyanyo	Bunyala	HaileS	Nyayo Hse	University	Westlands
1	8-Sep-14	Monday	KBX 192X	8:37:00 AM	8:43:00 AM	8:51:00 AM	8:58:00 AM	9:05:00 AM	9:13:00 AM
2	8-Sep-14	Monday	KBD 351F	8:43:00 AM	8:51:00 AM	8:58:00 AM	9:05:00 AM	9:13:00 AM	9:20:00 AM
3	8-Sep-14	Monday	KBL 462R	8:47:00 AM	8:55:00 AM	9:03:00 AM	9:11:00 AM	9:19:00 AM	9:25:00 AM
4	8-Sep-14	Monday	KBT 243B	8:52:00 AM	9:00:00 AM	9:08:00 AM	9:15:00 AM	9:23:00 AM	9:31:00 AM
5	8-Sep-14	Monday	KBJ 290R	9:08:00 AM	9:15:00 AM	9:24:00 AM	9:33:00 AM	9:41:00 AM	9:49:00 AM
6	8-Sep-14	Monday	KAR 936H	9:12:00 AM	9:22:00 AM	9:31:00 AM	9:39:00 AM	9:47:00 AM	9:58:00 AM
7	8-Sep-14	Monday	KBR 975R	9:13:00 AM	9:21:00 AM	9:29:00 AM	9:35:00 AM	9:40:00 AM	9:49:00 AM
8	8-Sep-14	Monday	KBZ 499E	9:13:00 AM	9:22:00 AM	9:31:00 AM	9:38:00 AM	9:42:00 AM	9:47:00 AM
9	8-Sep-14	Monday	KBW 130D	9:17:00 AM	9:25:00 AM	9:33:00 AM	9:40:00 AM	9:48:00 AM	9:57:00 AM
10	8-Sep-14	Monday	KBY 416A	9:20:00 AM	9:27:00 AM	9:35:00 AM	9:42:00 AM	9:49:00 AM	9:56:00 AM
11	8-Sep-14	Monday	KAP 848R	9:22:00 AM	9:30:00 AM	9:37:00 AM	9:44:00 AM	9:51:00 AM	9:59:00 AM
12	8-Sep-14	Monday	KBQ 450T	9:25:00 AM	9:31:00 AM	9:37:00 AM	9:42:00 AM	9:47:00 AM	9:54:00 AM
13	8-Sep-14	Monday	KBW 791K	9:27:00 AM	9:34:00 AM	9:42:00 AM	9:50:00 AM	9:57:00 AM	10:03:00 AM
14	8-Sep-14	Monday	KBT 005B	9:29:00 AM	9:37:00 AM	9:46:00 AM	9:54:00 AM	10:02:00 AM	10:11:00 AM
15	8-Sep-14	Monday	KBX 622P	9:29:00 AM	9:38:00 AM	9:46:00 AM	9:53:00 AM	10:01:00 AM	10:10:00 AM
16	8-Sep-14	Monday	KAV 841X	9:31:00 AM	9:36:00 AM	9:42:00 AM	9:47:00 AM	9:51:00 AM	9:55:00 AM
17	8-Sep-14	Monday	KBH 583M	9:31:00 AM	9:37:00 AM	9:43:00 AM	9:48:00 AM	9:54:00 AM	10:01:00 AM
18	8-Sep-14	Monday	KBL 958H	9:31:00 AM	9:38:00 AM	9:46:00 AM	9:53:00 AM	9:58:00 AM	10:04:00 AM
19	8-Sep-14	Monday	KBR 097R	9:31:00 AM	9:38:00 AM	9:44:00 AM	9:49:00 AM	9:52:00 AM	9:59:00 AM
20	8-Sep-14	Monday	KBU 447N	9:31:00 AM	9:38:00 AM	9:46:00 AM	9:53:00 AM	10:00:00 AM	10:09:00 AM
21	8-Sep-14	Monday	KBT 822X	9:33:00 AM	9:42:00 AM	9:51:00 AM	9:59:00 AM	10:05:00 AM	10:12:00 AM
22	8-Sep-14	Monday	KBA 164H	9:34:00 AM	9:41:00 AM	9:47:00 AM	9:54:00 AM	10:02:00 AM	10:10:00 AM
23	8-Sep-14	Monday	KAP 163C	9:35:00 AM	9:43:00 AM	9:51:00 AM	9:57:00 AM	10:04:00 AM	10:12:00 AM
24	8-Sep-14	Monday	KAX 086U	9:36:00 AM	9:45:00 AM	9:54:00 AM	10:02:00 AM	10:11:00 AM	10:20:00 AM
25	8-Sep-14	Monday	KBW 191E	9:36:00 AM	9:43:00 AM	9:51:00 AM	9:58:00 AM	10:04:00 AM	10:11:00 AM
26	8-Sep-14	Monday	KAE 804K	9:37:00 AM	9:45:00 AM	9:53:00 AM	10:00:00 AM	10:05:00 AM	10:11:00 AM
27	8-Sep-14	Monday	KBF 159N	9:38:00 AM	9:46:00 AM	9:54:00 AM	10:03:00 AM	10:10:00 AM	10:17:00 AM
28	8-Sep-14	Monday	KBP 780Q	9:40:00 AM	9:47:00 AM	9:54:00 AM	10:03:00 AM	10:12:00 AM	10:21:00 AM
29	8-Sep-14	Monday	KBW 639U	9:42:00 AM	9:50:00 AM	9:59:00 AM	10:04:00 AM	10:10:00 AM	10:18:00 AM
30	8-Sep-14	Monday	KBP 764N	9:45:00 AM	9:53:00 AM	10:02:00 AM	10:11:00 AM	10:19:00 AM	10:27:00 AM
31	8-Sep-14	Monday	KBR 236R	9:50:00 AM	9:58:00 AM	10:04:00 AM	10:11:00 AM	10:17:00 AM	10:24:00 AM
32	8-Sep-14	Monday	KBZ 208J	9:50:00 AM	9:56:00 AM	10:01:00 AM	10:07:00 AM	10:12:00 AM	10:16:00 AM

33	8-Sep-14	Monday	KBY 150Z	9:53:00 AM	10:00:00 AM	10:07:00 AM	10:13:00 AM	10:18:00 AM	10:23:00 AM
34	8-Sep-14	Monday	KAV 467H	9:54:00 AM	10:03:00 AM	10:11:00 AM	10:17:00 AM	10:25:00 AM	10:33:00 AM
35	8-Sep-14	Monday	KBJ 125X	9:55:00 AM	10:03:00 AM	10:14:00 AM	10:23:00 AM	10:33:00 AM	10:40:00 AM
36	8-Sep-14	Monday	KBY 079M	9:59:00 AM	10:04:00 AM	10:11:00 AM	10:19:00 AM	10:24:00 AM	10:30:00 AM
37	8-Sep-14	Monday	KBK 864X	10:00:00 AM	10:08:00 AM	10:13:00 AM	10:18:00 AM	10:24:00 AM	10:29:00 AM
38	8-Sep-14	Monday	KBY 958X	10:03:00 AM	10:10:00 AM	10:16:00 AM	10:22:00 AM	10:27:00 AM	10:33:00 AM
39	8-Sep-14	Monday	KBP 171Q	10:04:00 AM	10:11:00 AM	10:14:00 AM	10:18:00 AM	10:23:00 AM	10:28:00 AM
40	8-Sep-14	Monday	KAQ 085M	10:07:00 AM	10:14:00 AM	10:22:00 AM	10:27:00 AM	10:33:00 AM	10:41:00 AM
41	8-Sep-14	Monday	KAR 491T	10:09:00 AM	10:13:00 AM	10:18:00 AM	10:22:00 AM	10:25:00 AM	10:29:00 AM
42	8-Sep-14	Monday	KBR 578H	10:09:00 AM	10:13:00 AM	10:18:00 AM	10:22:00 AM	10:26:00 AM	10:31:00 AM
43	8-Sep-14	Monday	KBU 221E	10:11:00 AM	10:16:00 AM	10:20:00 AM	10:24:00 AM	10:29:00 AM	10:34:00 AM
44	8-Sep-14	Monday	KAN 089J	10:13:00 AM	10:17:00 AM	10:21:00 AM	10:26:00 AM	10:32:00 AM	10:37:00 AM
45	8-Sep-14	Monday	KBN 572A	10:14:00 AM	10:21:00 AM	10:25:00 AM	10:31:00 AM	10:36:00 AM	10:42:00 AM
46	8-Sep-14	Monday	KBR 346B	10:15:00 AM	10:22:00 AM	10:29:00 AM	10:34:00 AM	10:40:00 AM	10:46:00 AM
47	8-Sep-14	Monday	KBJ 336N	10:16:00 AM	10:23:00 AM	10:31:00 AM	10:35:00 AM	10:40:00 AM	10:46:00 AM
48	8-Sep-14	Monday	KBE 781G	10:17:00 AM	10:27:00 AM	10:35:00 AM	10:39:00 AM	10:44:00 AM	10:49:00 AM
49	8-Sep-14	Monday	KBJ 754K	10:17:00 AM	10:21:00 AM	10:27:00 AM	10:33:00 AM	10:39:00 AM	10:43:00 AM
50	8-Sep-14	Monday	KAW 387G	10:19:00 AM	10:27:00 AM	10:36:00 AM	10:44:00 AM	10:51:00 AM	11:00:00 AM

2. MSE RESULT FOR LEARNING CURVE

			Nyayo -Bunyala Section			
	ROOT MEAN SQUARE ERROR					
Data Train %	**RBF-KERNEL**		**LINEAR KERNEL**		**LINEAR REGRESSION**	
MSE	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
10%	0.06325	0.36472	0.06247	0.36624	0.06223	0.36484
20%	0.62376	0.26218	0.62341	0.26210	0.61748	0.26383
30%	0.56898	0.23409	0.56867	0.23397	0.56454	0.23464
40%	0.45467	0.25444	0.45450	0.25437	0.45138	0.25433
50%	0.46458	0.20459	0.46436	0.20454	0.46177	0.20435
60%	0.40756	0.22500	0.40744	0.22498	0.40505	0.22482
70%	0.38815	0.20982	0.38801	0.20969	0.38580	0.20956
80%	0.35145	0.26742	0.35134	0.26727	0.34946	0.26647
90%	0.36889	0.02732	0.36875	0.02727	0.36669	0.02892
95%	0.35113	0.02986	0.35061	0.02980	0.34875	0.03220
98%	0.34132	0.03049	0.34081	0.03034	0.33902	0.03179

3. MSE TRAINING AND TEST RESULTS

Nyayo Stadium to Westlands

		Mean Square Error(MSE)						
Data Train 90%	**Description**	**RBF-KERNEL**		**LINEAR KERNEL**		**LINEAR REGRESSION**		**RBF/Linear Kernel Average**
		Training Error	Test Error	Training Error	Test Error	Training Error	Test Error	Test Error
Section One	**Nyayo-Bunyala**	0.3688876	0.027241	0.3687508	0.02727019	0.36669343	0.028924925	0.02725576
Section Two	**Bunyala-Haile**	0.1167078	0.040673	0.1175452	0.04190400	0.11667994	0.041069661	0.04128874
Section Three	**Haile-Kenyatta Ave**	0.0755233	0.021561	0.0755422	0.02132496	0.07528952	0.021324968	0.02144341
Section Four	**Kenyatta Ave-University Way**	0.0851394	0.251914	0.0855015	0.25113414	0.08526778	0.249952784	0.25152408
Section Five	**University Way-Westlands**	4294992.2	6.259084	4294989.9	6.25770027	4293581.88	1387.496295	6.25839256

4. COEFFICIENT OF DETERMINATION (R^2)

Nyayo Stadium to Westlands

		Coefficient of Determination-R Squared								
Data Train 90%	**Description**	**RBF-KERNEL**			**LINEAR KERNEL**			**LINEAR REGRESSION**		
		Cross Validation	Training Accuracy	Test Accuracy	Cross Validation	Training Accuracy	Test Accuracy	Cross Validation	Training Accuracy	Test Accuracy
Section One	**Nyayo-Bunyala**	0.996636978	0.996606384	0.999737661	0.996638073	0.996607643	0.999737383	0.996652442	0.99662657	0.999721447
Section Two	**Bunyala-Haile**	0.998927446	0.998931751	0.999609599	0.998919365	0.998924087	0.999597788	0.998926984	0.998932007	0.999605797
Section Three	**Haile-Kenyatta Ave**	0.99930213	0.999313681	0.999787428	0.999302151	0.999313509	0.999796449	0.999304331	0.999315806	0.999793142
Section Four	**Kenyatta Ave-University Way**	0.999235724	0.999232102	0.997622042	0.999232533	0.999228836	0.997629403	0.999234085	0.999230944	0.997640555
Section Five	**University Way-Westlands**	0.860998382	-0.000328046	0.941654438	0.861043729	-0.000327503	0.941770685	-12.84148066	4.37E-07	-11.91096657

4. PARTITIONED DATA RESULTS (R^2)

		R-Squared(8-11am)			
Data Train 90%	Description	RBF-KERNEL		LINEAR KERNEL	LINEAR KERNEL
		Training Accuracy	Test Accuracy	Training Accuracy	Test Accuracy
Section One	Nyayo-Bunyala	0.998055232	0.99796829	0.998034506	0.997885604
Section Five	University Way-Westlands	0.838248379	0.912259865	0.837987454	0.911823906

		R-Squared(11-3pm)			
Data Train 90%	Description	RBF-KERNEL		LINEAR KERNEL	LINEAR KERNEL
		Training Accuracy	Test Accuracy	Training Accuracy	Test Accuracy
Section One	Nyayo-Bunyala	0.973801832	0.995142675	0.973799903	0.995149794
Section Five	University Way-Westlands	-0.000699119	0.43004147	0.429887052	-0.000699032

		R-Squared(3-5pm)			
Data Train 90%	Description	RBF-KERNEL		LINEAR KERNEL	LINEAR KERNEL
		Training Accuracy	Test Accuracy	Training Accuracy	Test Accuracy
Section One	Nyayo-Bunyala	0.9882841	0.988665107	0.988197207	0.988626089
Section Five	University Way-Westlands	0.48857592	0.161735927	0.488526647	0.161747286

		R-Squared(Average)			
Data Train 90%	Description	RBF-KERNEL		LINEAR KERNEL	
		Training Accuracy	Test Accuracy	Training Accuracy	Test Accuracy
Section One	Nyayo-Bunyala	0.9867	0.9939	0.9867	0.9939
Section Five	University Way-Westlands	0.4420	0.5013	0.5855	0.3576

APPENDIX B: PREDICTIONS RESULTS

1. SECTION ONE

No	Nyayo	Bunyala	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	3:29:00 PM	3:37:00 PM	3:35:36 PM	3:35:32 PM	3:36:23 PM
2	3:11:00 PM	3:16:00 PM	3:17:34 PM	3:17:31 PM	3:18:21 PM
3	1:10:00 PM	1:15:00 PM	1:16:22 PM	1:16:26 PM	1:17:07 PM
4	1:19:00 PM	1:24:00 PM	1:25:23 PM	1:25:27 PM	1:26:08 PM
5	2:56:00 PM	3:01:00 PM	3:02:32 PM	3:02:30 PM	3:03:20 PM
6	2:06:00 PM	2:10:00 PM	2:12:26 PM	2:12:28 PM	2:13:14 PM
7	1:39:00 PM	1:45:00 PM	1:45:24 PM	1:45:27 PM	1:46:10 PM
8	10:56:00 AM	11:00:00 AM	11:02:24 AM	11:02:21 AM	11:02:51 AM
9	11:01:00 AM	11:10:00 AM	11:07:24 AM	11:07:21 AM	11:07:52 AM
10	1:52:00 PM	2:00:00 PM	1:58:25 PM	1:58:28 PM	1:59:12 PM
11	2:44:00 PM	2:56:00 PM	2:50:30 PM	2:50:30 PM	2:51:18 PM
12	12:18:00 PM	12:25:00 PM	12:24:22 PM	12:24:24 PM	12:25:01 PM
13	1:14:00 PM	1:22:00 PM	1:20:23 PM	1:20:26 PM	1:21:07 PM
14	1:22:00 PM	1:28:00 PM	1:28:23 PM	1:28:27 PM	1:29:08 PM
15	7:05:00 AM	7:11:00 AM	7:11:25 AM	7:11:12 AM	7:11:24 AM
16	3:33:00 PM	3:39:00 PM	3:39:37 PM	3:39:32 PM	3:40:24 PM
17	8:26:00 AM	8:32:00 AM	8:32:27 AM	8:32:15 AM	8:32:33 AM
18	12:24:00 PM	12:29:00 PM	12:30:22 PM	12:30:24 PM	12:31:02 PM
19	1:02:00 PM	1:10:00 PM	1:08:22 PM	1:08:26 PM	1:09:06 PM
20	2:44:00 PM	2:49:00 PM	2:50:30 PM	2:50:30 PM	2:51:18 PM
21	1:46:00 PM	1:54:00 PM	1:52:24 PM	1:52:28 PM	1:53:11 PM
22	10:24:00 AM	10:33:00 AM	10:30:25 AM	10:30:20 AM	10:30:47 AM
23	11:36:00 AM	11:41:00 AM	11:42:23 AM	11:42:23 AM	11:42:56 AM
24	11:55:00 AM	12:03:00 PM	12:01:22 PM	12:01:23 PM	12:01:58 PM
25	9:57:00 AM	10:06:00 AM	10:03:26 AM	10:03:19 AM	10:03:44 AM
26	9:55:00 AM	10:01:00 AM	10:01:26 AM	10:01:19 AM	10:01:44 AM
27	9:13:00 AM	9:18:00 AM	9:19:27 AM	9:19:17 AM	9:19:39 AM
28	11:54:00 AM	11:59:00 AM	12:00:22 PM	12:00:23 PM	12:00:58 PM
29	4:20:00 PM	4:29:00 PM	4:26:42 PM	4:26:33 PM	4:27:29 PM
30	10:11:00 AM	10:13:00 AM	10:17:26 AM	10:17:19 AM	10:17:46 AM
31	10:44:00 AM	10:48:00 AM	10:50:24 AM	10:50:21 AM	10:50:50 AM
32	2:30:00 PM	2:37:00 PM	2:36:29 PM	2:36:29 PM	2:37:16 PM
33	7:43:00 AM	7:44:00 AM	7:49:26 AM	7:49:14 AM	7:49:28 AM
34	12:14:00 PM	12:17:00 PM	12:20:22 PM	12:20:24 PM	12:21:00 PM
35	10:15:00 AM	10:22:00 AM	10:21:26 AM	10:21:20 AM	10:21:46 AM

36	6:46:00 AM	6:52:00 AM	6:52:24 AM	6:52:12 AM	6:52:22 AM
37	1:49:00 PM	1:58:00 PM	1:55:25 PM	1:55:28 PM	1:56:12 PM
38	9:54:00 AM	10:00:00 AM	10:00:26 AM	10:00:19 AM	10:00:44 AM
39	11:29:00 AM	11:35:00 AM	11:35:23 AM	11:35:22 AM	11:35:55 AM
40	3:58:00 PM	4:07:00 PM	4:04:40 PM	4:04:33 PM	4:05:27 PM
41	3:34:00 PM	3:46:00 PM	3:40:37 PM	3:40:32 PM	3:41:24 PM
42	2:49:00 PM	2:54:00 PM	2:55:31 PM	2:55:30 PM	2:56:19 PM
43	9:13:00 AM	9:21:00 AM	9:19:27 AM	9:19:17 AM	9:19:39 AM
44	1:38:00 PM	1:42:00 PM	1:44:24 PM	1:44:27 PM	1:45:10 PM
45	9:04:00 AM	9:13:00 AM	9:10:27 AM	9:10:17 AM	9:10:38 AM
46	11:39:00 AM	11:47:00 AM	11:45:22 AM	11:45:23 AM	11:45:56 AM
47	1:35:00 PM	1:42:00 PM	1:41:24 PM	1:41:27 PM	1:42:10 PM
48	10:58:00 AM	11:05:00 AM	11:04:24 AM	11:04:21 AM	11:04:51 AM
49	2:51:00 PM	3:05:00 PM	2:57:31 PM	2:57:30 PM	2:58:19 PM
50	2:44:00 PM	2:56:00 PM	2:50:30 PM	2:50:30 PM	2:51:18 PM

2. SECTION TWO

No	Bunyala	Haile Selassie	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	3:37:00 PM	3:46:00 PM	3:44:04 PM	3:43:36 PM	3:44:14 PM
2	3:16:00 PM	3:21:00 PM	3:22:53 PM	3:22:35 PM	3:23:11 PM
3	1:15:00 PM	1:25:00 PM	1:21:20 PM	1:21:31 PM	1:21:57 PM
4	1:24:00 PM	1:31:00 PM	1:30:21 PM	1:30:31 PM	1:30:58 PM
5	3:01:00 PM	3:06:00 PM	3:07:46 PM	3:07:35 PM	3:08:09 PM
6	2:10:00 PM	2:13:00 PM	2:16:29 PM	2:16:33 PM	2:17:03 PM
7	1:45:00 PM	1:56:00 PM	1:51:24 PM	1:51:32 PM	1:52:00 PM
8	11:00:00 AM	11:07:00 AM	11:06:25 AM	11:06:26 AM	11:06:40 AM
9	11:10:00 AM	11:18:00 AM	11:16:24 AM	11:16:26 AM	11:16:42 AM
10	2:00:00 PM	2:07:00 PM	2:06:27 PM	2:06:33 PM	2:07:02 PM
11	2:56:00 PM	3:03:00 PM	3:02:44 PM	3:02:35 PM	3:03:09 PM
12	12:25:00 PM	12:32:00 PM	12:31:19 PM	12:31:29 PM	12:31:51 PM
13	1:22:00 PM	1:31:00 PM	1:28:21 PM	1:28:31 PM	1:28:57 PM
14	1:28:00 PM	1:34:00 PM	1:34:21 PM	1:34:31 PM	1:34:58 PM
15	7:11:00 AM	7:16:00 AM	7:17:23 AM	7:17:18 AM	7:17:13 AM
16	3:39:00 PM	3:44:00 PM	3:46:05 PM	3:45:36 PM	3:46:14 PM
17	8:32:00 AM	8:37:00 AM	8:38:31 AM	8:38:21 AM	8:38:23 AM
18	12:29:00 PM	12:34:00 PM	12:35:19 PM	12:35:29 PM	12:35:51 PM
19	1:10:00 PM	1:17:00 PM	1:16:20 PM	1:16:31 PM	1:16:56 PM
20	2:49:00 PM	2:52:00 PM	2:55:41 PM	2:55:34 PM	2:56:08 PM
21	1:54:00 PM	2:03:00 PM	2:00:26 PM	2:00:32 PM	2:01:01 PM

22	10:33:00 AM	10:41:00 AM	10:39:27 AM	10:39:25 AM	10:39:37 AM
23	11:41:00 AM	11:52:00 AM	11:47:21 AM	11:47:28 AM	11:47:45 AM
24	12:03:00 PM	12:11:00 PM	12:09:20 PM	12:09:28 PM	12:09:48 PM
25	10:06:00 AM	10:13:00 AM	10:12:29 AM	10:12:24 AM	10:12:34 AM
26	10:01:00 AM	10:12:00 AM	10:07:30 AM	10:07:24 AM	10:07:33 AM
27	9:18:00 AM	9:27:00 AM	9:24:32 AM	9:24:22 AM	9:24:28 AM
28	11:59:00 AM	12:03:00 PM	12:05:20 PM	12:05:28 PM	12:05:47 PM
29	4:29:00 PM	4:38:00 PM	4:36:39 PM	4:35:38 PM	4:36:20 PM
30	10:13:00 AM	10:18:00 AM	10:19:29 AM	10:19:24 AM	10:19:35 AM
31	10:48:00 AM	10:53:00 AM	10:54:26 AM	10:54:26 AM	10:54:39 AM
32	2:37:00 PM	2:43:00 PM	2:43:37 PM	2:43:34 PM	2:44:06 PM
33	7:44:00 AM	7:53:00 AM	7:50:27 AM	7:50:19 AM	7:50:17 AM
34	12:17:00 PM	12:21:00 PM	12:23:19 PM	12:23:29 PM	12:23:50 PM
35	10:22:00 AM	10:29:00 AM	10:28:28 AM	10:28:25 AM	10:28:36 AM
36	6:52:00 AM	6:59:00 AM	6:58:20 AM	6:58:17 AM	6:58:11 AM
37	1:58:00 PM	2:04:00 PM	2:04:26 PM	2:04:33 PM	2:05:02 PM
38	10:00:00 AM	10:07:00 AM	10:06:30 AM	10:06:24 AM	10:06:33 AM
39	11:35:00 AM	11:39:00 AM	11:41:22 AM	11:41:27 AM	11:41:45 AM
40	4:07:00 PM	4:15:00 PM	4:14:23 PM	4:13:37 PM	4:14:17 PM
41	3:46:00 PM	3:54:00 PM	3:53:09 PM	3:52:37 PM	3:53:15 PM
42	2:54:00 PM	3:01:00 PM	3:00:43 PM	3:00:35 PM	3:01:08 PM
43	9:21:00 AM	9:29:00 AM	9:27:32 AM	9:27:22 AM	9:27:29 AM
44	1:42:00 PM	1:45:00 PM	1:48:23 PM	1:48:32 PM	1:49:00 PM
45	9:13:00 AM	9:20:00 AM	9:19:32 AM	9:19:22 AM	9:19:28 AM
46	11:47:00 AM	11:54:00 AM	11:53:21 AM	11:53:28 AM	11:53:46 AM
47	1:42:00 PM	1:48:00 PM	1:48:23 PM	1:48:32 PM	1:49:00 PM
48	11:05:00 AM	11:07:00 AM	11:11:24 AM	11:11:26 AM	11:11:41 AM
49	3:05:00 PM	3:08:00 PM	3:11:48 PM	3:11:35 PM	3:12:10 PM
50	2:56:00 PM	3:03:00 PM	3:02:44 PM	3:02:35 PM	3:03:09 PM

3. SECTION 3

No	Haile Selassie	Kenyatta Avenue	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	3:46:00 PM	3:54:00 PM	3:53:03 PM	3:52:44 PM	3:53:02 PM
2	3:21:00 PM	3:27:00 PM	3:27:49 PM	3:27:40 PM	3:27:57 PM
3	1:25:00 PM	1:30:00 PM	1:31:02 PM	1:31:20 PM	1:31:34 PM
4	1:31:00 PM	1:39:00 PM	1:37:04 PM	1:37:21 PM	1:37:35 PM
5	3:06:00 PM	3:11:00 PM	3:12:42 PM	3:12:37 PM	3:12:54 PM
6	2:13:00 PM	2:20:00 PM	2:19:18 PM	2:19:28 PM	2:19:43 PM
7	1:56:00 PM	2:01:00 PM	2:02:12 PM	2:02:25 PM	2:02:40 PM

8	11:07:00 AM	11:09:00 AM	11:12:38 AM	11:12:55 AM	11:13:07 AM
9	11:18:00 AM	11:25:00 AM	11:23:39 AM	11:23:57 AM	11:24:09 AM
10	2:07:00 PM	2:16:00 PM	2:13:16 PM	2:13:27 PM	2:13:42 PM
11	3:03:00 PM	3:08:00 PM	3:09:40 PM	3:09:37 PM	3:09:53 PM
12	12:32:00 PM	12:39:00 PM	12:37:50 PM	12:38:10 PM	12:38:24 PM
13	1:31:00 PM	1:38:00 PM	1:37:04 PM	1:37:21 PM	1:37:35 PM
14	1:34:00 PM	1:39:00 PM	1:40:05 PM	1:40:21 PM	1:40:36 PM
15	7:16:00 AM	7:22:00 AM	7:21:46 AM	7:21:15 AM	7:21:22 AM
16	3:44:00 PM	3:50:00 PM	3:51:02 PM	3:50:44 PM	3:51:01 PM
17	8:37:00 AM	8:41:00 AM	8:42:34 AM	8:42:29 AM	8:42:38 AM
18	12:34:00 PM	12:39:00 PM	12:39:50 PM	12:40:11 PM	12:40:24 PM
19	1:17:00 PM	1:24:00 PM	1:23:00 PM	1:23:18 PM	1:23:32 PM
20	2:52:00 PM	2:57:00 PM	2:58:35 PM	2:58:35 PM	2:58:51 PM
21	2:03:00 PM	2:12:00 PM	2:09:15 PM	2:09:26 PM	2:09:41 PM
22	10:41:00 AM	10:49:00 AM	10:46:35 AM	10:46:51 AM	10:47:02 AM
23	11:52:00 AM	11:58:00 AM	11:57:43 AM	11:58:03 AM	11:58:16 AM
24	12:11:00 PM	12:19:00 PM	12:16:46 PM	12:17:07 PM	12:17:19 PM
25	10:13:00 AM	10:18:00 AM	10:18:34 AM	10:18:46 AM	10:18:56 AM
26	10:12:00 AM	10:21:00 AM	10:17:34 AM	10:17:46 AM	10:17:56 AM
27	9:27:00 AM	9:31:00 AM	9:32:33 AM	9:32:38 AM	9:32:47 AM
28	12:03:00 PM	12:10:00 PM	12:08:44 PM	12:09:05 PM	12:09:18 PM
29	4:38:00 PM	4:49:00 PM	4:45:35 PM	4:44:54 PM	4:45:12 PM
30	10:18:00 AM	10:22:00 AM	10:23:34 AM	10:23:47 AM	10:23:57 AM
31	10:53:00 AM	10:58:00 AM	10:58:36 AM	10:58:53 AM	10:59:04 AM
32	2:43:00 PM	2:46:00 PM	2:49:31 PM	2:49:33 PM	2:49:49 PM
33	7:53:00 AM	7:58:00 AM	7:58:38 AM	7:58:21 AM	7:58:29 AM
34	12:21:00 PM	12:23:00 PM	12:26:47 PM	12:27:08 PM	12:27:21 PM
35	10:29:00 AM	10:34:00 AM	10:34:35 AM	10:34:49 AM	10:34:59 AM
36	6:59:00 AM	7:04:00 AM	7:04:50 AM	7:04:12 AM	7:04:18 AM
37	2:04:00 PM	2:10:00 PM	2:10:15 PM	2:10:26 PM	2:10:42 PM
38	10:07:00 AM	10:13:00 AM	10:12:34 AM	10:12:45 AM	10:12:55 AM
39	11:39:00 AM	11:45:00 AM	11:44:41 AM	11:45:01 AM	11:45:13 AM
40	4:15:00 PM	4:24:00 PM	4:22:20 PM	4:21:50 PM	4:22:07 PM
41	3:54:00 PM	4:06:00 PM	4:01:08 PM	4:00:46 PM	4:01:03 PM
42	3:01:00 PM	3:09:00 PM	3:07:39 PM	3:07:37 PM	3:07:53 PM
43	9:29:00 AM	9:37:00 AM	9:34:33 AM	9:34:38 AM	9:34:48 AM
44	1:45:00 PM	1:48:00 PM	1:51:08 PM	1:51:23 PM	1:51:38 PM
45	9:20:00 AM	9:27:00 AM	9:25:33 AM	9:25:37 AM	9:25:46 AM
46	11:54:00 AM	12:02:00 PM	11:59:43 AM	12:00:04 PM	12:00:16 PM
47	1:48:00 PM	1:53:00 PM	1:54:09 PM	1:54:24 PM	1:54:38 PM
48	11:07:00 AM	11:11:00 AM	11:12:38 AM	11:12:55 AM	11:13:07 AM

49	3:08:00 PM	3:16:00 PM	3:14:43 PM	3:14:38 PM	3:14:54 PM
50	3:03:00 PM	3:09:00 PM	3:09:40 PM	3:09:37 PM	3:09:53 PM

4. SECTION 4

No	Kenyatta Avenue	University Way	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	3:54:00 PM	4:00:00 PM	4:00:42 PM	4:00:27 PM	4:00:45 PM
2	3:27:00 PM	3:33:00 PM	3:33:23 PM	3:33:22 PM	3:33:40 PM
3	1:30:00 PM	1:36:00 PM	1:35:35 PM	1:36:02 PM	1:36:16 PM
4	1:39:00 PM	1:46:00 PM	1:44:37 PM	1:45:04 PM	1:45:18 PM
5	3:11:00 PM	3:20:00 PM	3:17:13 PM	3:17:20 PM	3:17:36 PM
6	2:20:00 PM	2:25:00 PM	2:25:49 PM	2:26:11 PM	2:26:26 PM
7	2:01:00 PM	2:07:00 PM	2:06:43 PM	2:07:08 PM	2:07:22 PM
8	11:09:00 AM	11:18:00 AM	11:14:27 AM	11:14:38 AM	11:14:47 AM
9	11:25:00 AM	11:28:00 AM	11:30:27 AM	11:30:40 AM	11:30:50 AM
10	2:16:00 PM	2:25:00 PM	2:21:48 PM	2:22:10 PM	2:22:25 PM
11	3:08:00 PM	3:13:00 PM	3:14:11 PM	3:14:19 PM	3:14:36 PM
12	12:39:00 PM	12:44:00 PM	12:44:28 PM	12:44:53 PM	12:45:05 PM
13	1:38:00 PM	1:45:00 PM	1:43:36 PM	1:44:03 PM	1:44:17 PM
14	1:39:00 PM	1:46:00 PM	1:44:37 PM	1:45:04 PM	1:45:18 PM
15	7:22:00 AM	7:28:00 AM	7:27:34 AM	7:26:58 AM	7:27:01 AM
16	3:50:00 PM	3:54:00 PM	3:56:39 PM	3:56:26 PM	3:56:44 PM
17	8:41:00 AM	8:45:00 AM	8:46:33 AM	8:46:12 AM	8:46:17 AM
18	12:39:00 PM	12:46:00 PM	12:44:28 PM	12:44:53 PM	12:45:05 PM
19	1:24:00 PM	1:30:00 PM	1:29:33 PM	1:30:01 PM	1:30:15 PM
20	2:57:00 PM	3:01:00 PM	3:03:05 PM	3:03:17 PM	3:03:33 PM
21	2:12:00 PM	2:20:00 PM	2:17:46 PM	2:18:09 PM	2:18:24 PM
22	10:49:00 AM	10:54:00 AM	10:54:28 AM	10:54:34 AM	10:54:43 AM
23	11:58:00 AM	11:59:00 AM	12:03:26 PM	12:03:46 PM	12:03:57 PM
24	12:19:00 PM	12:26:00 PM	12:24:27 PM	12:24:50 PM	12:25:01 PM
25	10:18:00 AM	10:25:00 AM	10:23:29 AM	10:23:29 AM	10:23:37 AM
26	10:21:00 AM	10:30:00 AM	10:26:29 AM	10:26:29 AM	10:26:37 AM
27	9:31:00 AM	9:34:00 AM	9:36:31 AM	9:36:20 AM	9:36:27 AM
28	12:10:00 PM	12:14:00 PM	12:15:26 PM	12:15:48 PM	12:16:00 PM
29	4:49:00 PM	5:00:00 PM	4:56:30 PM	4:55:37 PM	4:55:56 PM
30	10:22:00 AM	10:24:00 AM	10:27:29 AM	10:27:29 AM	10:27:38 AM
31	10:58:00 AM	11:04:00 AM	11:03:28 AM	11:03:36 AM	11:03:45 AM
32	2:46:00 PM	2:53:00 PM	2:52:00 PM	2:52:15 PM	2:52:31 PM
33	7:58:00 AM	8:04:00 AM	8:03:33 AM	8:03:04 AM	8:03:08 AM
34	12:23:00 PM	12:26:00 PM	12:28:27 PM	12:28:50 PM	12:29:02 PM

35	10:34:00 AM	10:40:00 AM	10:39:29 AM	10:39:31 AM	10:39:40 AM
36	7:04:00 AM	7:10:00 AM	7:09:35 AM	7:08:55 AM	7:08:57 AM
37	2:10:00 PM	2:16:00 PM	2:15:45 PM	2:16:09 PM	2:16:24 PM
38	10:13:00 AM	10:17:00 AM	10:18:30 AM	10:18:28 AM	10:18:36 AM
39	11:45:00 AM	11:52:00 AM	11:50:26 AM	11:50:44 AM	11:50:54 AM
40	4:24:00 PM	4:32:00 PM	4:31:07 PM	4:30:32 PM	4:30:51 PM
41	4:06:00 PM	4:18:00 PM	4:12:51 PM	4:12:29 PM	4:12:48 PM
42	3:09:00 PM	3:15:00 PM	3:15:12 PM	3:15:19 PM	3:15:36 PM
43	9:37:00 AM	9:44:00 AM	9:42:31 AM	9:42:22 AM	9:42:28 AM
44	1:48:00 PM	1:53:00 PM	1:53:39 PM	1:54:05 PM	1:54:19 PM
45	9:27:00 AM	9:33:00 AM	9:32:32 AM	9:32:20 AM	9:32:26 AM
46	12:02:00 PM	12:11:00 PM	12:07:26 PM	12:07:47 PM	12:07:58 PM
47	1:53:00 PM	2:00:00 PM	1:58:40 PM	1:59:06 PM	1:59:20 PM
48	11:11:00 AM	11:15:00 AM	11:16:27 AM	11:16:38 AM	11:16:48 AM
49	3:16:00 PM	3:22:00 PM	3:22:16 PM	3:22:21 PM	3:22:37 PM
50	3:09:00 PM	3:13:00 PM	3:15:12 PM	3:15:19 PM	3:15:36 PM

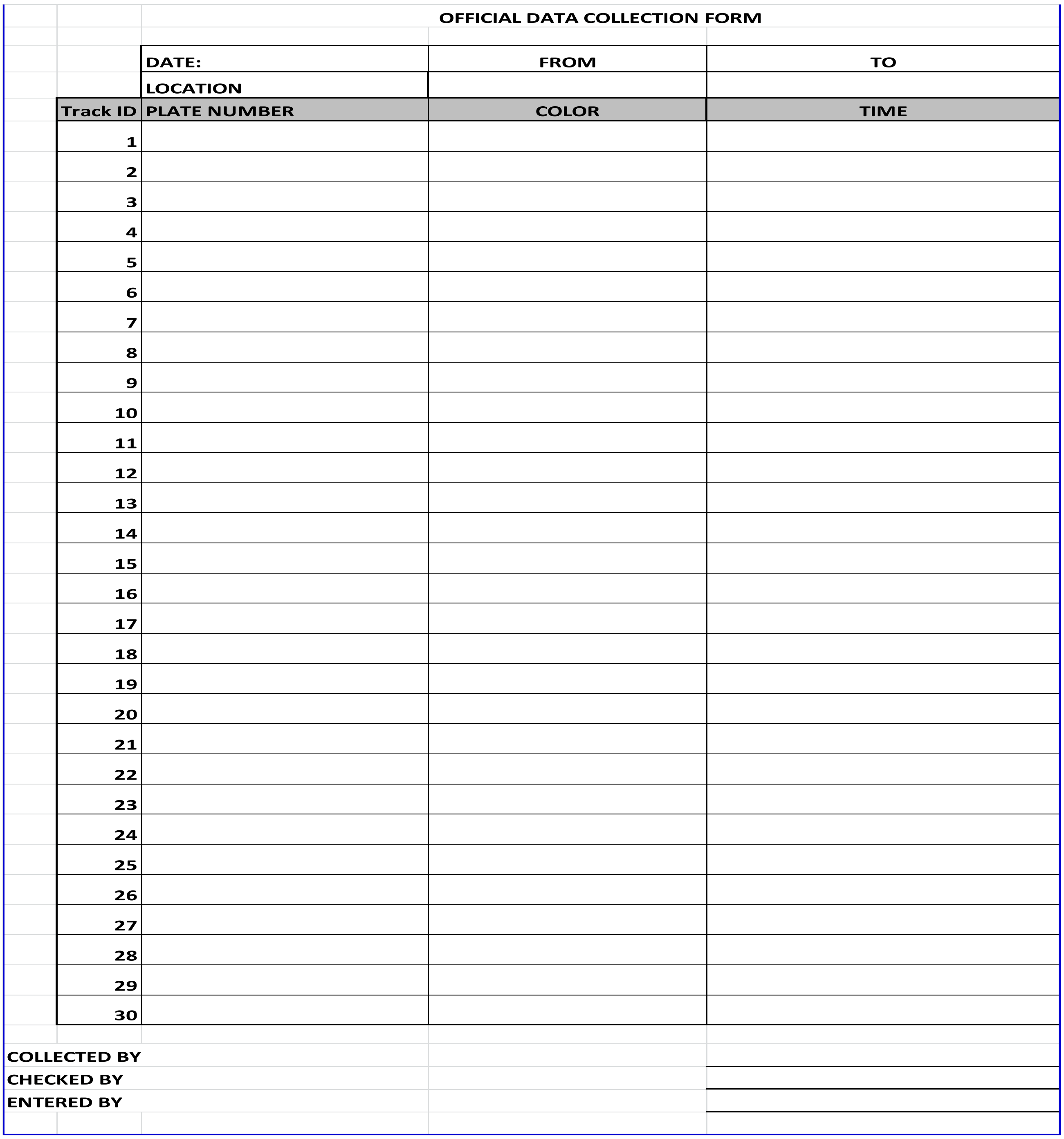
5. SECTION 5

No	University Way	Westlands	RBF Kernel	Linear Kernel	Linear Regression
	Time In	Actual Time Out	Predicted	Predicted	Predicted
1	4:00:00 PM	4:11:00 PM	4:08:56 PM	4:08:40 PM	9:09:32 PM
2	3:33:00 PM	3:40:00 PM	3:41:33 PM	3:41:36 PM	9:13:03 PM
3	1:36:00 PM	1:50:00 PM	1:43:36 PM	1:44:15 PM	9:28:16 PM
4	1:46:00 PM	1:53:00 PM	1:53:38 PM	1:54:16 PM	9:26:58 PM
5	3:20:00 PM	3:27:00 PM	3:28:23 PM	3:28:33 PM	9:14:44 PM
6	2:25:00 PM	2:33:00 PM	2:32:51 PM	2:33:23 PM	9:21:53 PM
7	2:07:00 PM	2:16:00 PM	2:14:44 PM	2:15:20 PM	9:24:14 PM
8	11:18:00 AM	11:25:00 AM	11:25:33 AM	11:25:50 AM	9:46:13 PM
9	11:28:00 AM	11:34:00 AM	11:35:32 AM	11:35:52 AM	9:44:55 PM
10	2:25:00 PM	2:35:00 PM	2:32:51 PM	2:33:23 PM	9:21:53 PM
11	3:13:00 PM	3:21:00 PM	3:21:18 PM	3:21:32 PM	9:15:39 PM
12	12:44:00 PM	12:52:00 PM	12:51:29 PM	12:52:05 PM	9:35:02 PM
13	1:45:00 PM	1:53:00 PM	1:52:38 PM	1:53:16 PM	9:27:06 PM
14	1:46:00 PM	1:55:00 PM	1:53:38 PM	1:54:16 PM	9:26:58 PM
15	7:28:00 AM	7:35:00 AM	7:35:14 AM	7:35:08 AM	10:16:08 PM
16	3:54:00 PM	4:01:00 PM	4:02:51 PM	4:02:39 PM	9:10:19 PM
17	8:45:00 AM	8:50:00 AM	8:52:34 AM	8:52:22 AM	10:06:07 PM
18	12:46:00 PM	12:55:00 PM	12:53:29 PM	12:54:06 PM	9:34:46 PM
19	1:30:00 PM	1:37:00 PM	1:37:34 PM	1:38:13 PM	9:29:03 PM
20	3:01:00 PM	3:07:00 PM	3:09:10 PM	3:09:30 PM	9:17:12 PM

21	2:20:00 PM	2:29:00 PM	2:27:49 PM	2:28:22 PM	9:22:32 PM
22	10:54:00 AM	11:01:00 AM	11:01:35 AM	11:01:45 AM	9:49:20 PM
23	11:59:00 AM	12:12:00 PM	12:06:30 PM	12:06:57 PM	9:40:53 PM
24	12:26:00 PM	12:33:00 PM	12:33:29 PM	12:34:02 PM	9:37:22 PM
25	10:25:00 AM	10:32:00 AM	10:32:37 AM	10:32:40 AM	9:53:07 PM
26	10:30:00 AM	10:38:00 AM	10:37:37 AM	10:37:41 AM	9:52:28 PM
27	9:34:00 AM	9:44:00 AM	9:41:38 AM	9:41:31 AM	9:59:45 PM
28	12:14:00 PM	12:19:00 PM	12:21:29 PM	12:22:00 PM	9:38:56 PM
29	5:00:00 PM	5:09:00 PM	5:10:03 PM	5:08:51 PM	9:01:44 PM
30	10:24:00 AM	10:40:00 AM	10:31:37 AM	10:31:40 AM	9:53:14 PM
31	11:04:00 AM	11:09:00 AM	11:11:34 AM	11:11:47 AM	9:48:02 PM
32	2:53:00 PM	2:59:00 PM	3:01:05 PM	3:01:28 PM	9:18:15 PM
33	8:04:00 AM	8:11:00 AM	8:11:26 AM	8:11:15 AM	10:11:27 PM
34	12:26:00 PM	12:30:00 PM	12:33:29 PM	12:34:02 PM	9:37:22 PM
35	10:40:00 AM	10:46:00 AM	10:47:36 AM	10:47:43 AM	9:51:09 PM
36	7:10:00 AM	7:17:00 AM	7:17:07 AM	7:17:05 AM	10:18:28 PM
37	2:16:00 PM	2:22:00 PM	2:23:47 PM	2:24:22 PM	9:23:04 PM
38	10:17:00 AM	10:27:00 AM	10:24:37 AM	10:24:39 AM	9:54:09 PM
39	11:52:00 AM	11:59:00 AM	11:59:30 AM	11:59:56 AM	9:41:48 PM
40	4:32:00 PM	4:44:00 PM	4:41:29 PM	4:40:46 PM	9:05:22 PM
41	4:18:00 PM	4:29:00 PM	4:27:14 PM	4:26:44 PM	9:07:12 PM
42	3:15:00 PM	3:22:00 PM	3:23:19 PM	3:23:32 PM	9:15:23 PM
43	9:44:00 AM	9:51:00 AM	9:51:38 AM	9:51:33 AM	9:58:27 PM
44	1:53:00 PM	2:00:00 PM	2:00:40 PM	2:01:18 PM	9:26:03 PM
45	9:33:00 AM	9:39:00 AM	9:40:38 AM	9:40:31 AM	9:59:52 PM
46	12:11:00 PM	12:20:00 PM	12:18:29 PM	12:18:59 PM	9:39:19 PM
47	2:00:00 PM	2:09:00 PM	2:07:42 PM	2:08:19 PM	9:25:09 PM

APPENDIX C: DATA COLLECTION TOOLS

1. DATA COLLECTION FORM



OFFICIAL DATA COLLECTION FORM

	DATE:	FROM	TO
	LOCATION		
Track ID	PLATE NUMBER	COLOR	TIME
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			

COLLECTED BY

CHECKED BY

ENTERED BY

2. INTERVIEW QUESTIONNAIRE

DATA CLERK INTERVIEW QUESTIONS

Please note the number indicated above and then answer the following questions as frankly as possible:-

This research study involves gathering traffic data for trucks passing Westlands round-about to Nyayo stadium round-about. This shall involve recording specified information on a predefined form that will be provided on a daily basis. It is expected of you to carry out this process in an honest manner.

1. How available are you from 8th September to 30th September 2014?
2. Are you comfortable working from 8 a.m. to 5 p.m. recording similar data for different objects you observed?
3. How do you motivate yourself when performing routine tasks?
4. Do you have a challenge describing an object using color and size?
5. Do you have a problem interacting with traffic policemen?
6. Are you comfortable commuting to town daily and being paid on a weekly basis?

3. CONTRACT OF EMPLOYMENT

Entered into between: **<u>Brian Otieno Onyango</u>** (herein after referred to as "the employer") **Address of employer**: ……………………………………………………………. ……………………………………………………………. ……………………………………………………………. …………………………………………………………….	**And** ……………………………………………………………. (Herein after referred to as "the employee") **Address of employer**: ……………………………………………………………. ……………………………………………………………. ……………………………………………………………. …………………………………………………………….

1. Commencement

This contract will begin on …………….. and shall run for an initial period of **two (2) calendar weeks** with a possibility of **one(1) week** extension until terminated as set out in clause 4.

2. Place of work

……………………………………………………

3. Job description

Job Title

……………………………………………….
(Research Assistant)

REPRESENTATIVE DUTIES:

1. **Data Collection**: Collects various forms of data pertaining to the research project or projects.

2. **Record Keeping**: Keeps record of information obtained during research. May include a database of information, hard files

3. Hand over completed data form to the Head of Research assistants

4. Carry a copy of the research permit and the relevant work documents as provided for by the head of Research.

5. **As Needed.** Performs various duties as needed to successfully fulfill the function of the position

4. Termination of employment

Either party can terminate this agreement with one (1) day notice. In the case where an employee has willingly and knowingly tampered with data, notice may be given to that employee verbally and the contract cancelled with immediate effect.

5. Wage

5.1	The employees wage shall be paid in cash on the last working day of every week and shall be:	**KES 500** per day
5.2	The employee shall be entitled to good performance allowance	**10%** Per week
5.3	Umbrella Allowance(If need be)	**KES 300/-**
5.4	The total value of the above remuneration shall be	KES…………
	(The total of clauses 5.1 to 5.3)	

6. Hours of work

6.1 Normal working hours will be from ……….. a.m. to ………… p.m. on Mondays to Fridays and from ……………a.m. to ………..p.m. on Saturdays.

7. Absenteeism

The employer will deduct an amount equal to one day's wage if the employee does not report to work without giving not less than12 hour's prior verbal notice to the head of research.

8. Meal Intervals

The employee agrees to a lunch break for Thirty (30) minutes. Lunchtime will be taken from ………. to …………… daily unless instructed otherwise.

9. Deductions from remuneration

The employer may not deduct any monies from the employee's wage unless the employee has agreed to this in writing on each occasion.

10. General

Any changes to this agreement will only be valid if they are in writing and have been agreed and signed by both parties.

Please Note: These guidelines are not meant to be a complete summary of the Basic Conditions of Employment Act and/or legal advice. Should there be any doubt as to rights and/or obligations in terms of the Act or terms of any clause of the suggested Contract of Employment, such queries can be directed to the local office of the Department of Labor, who will gladly assist

THUS DONE AND SIGNED AT …………………….……ON THIS ……….. DAY OF………………….. 2014

EMPLOYER **EMPLOYEE**

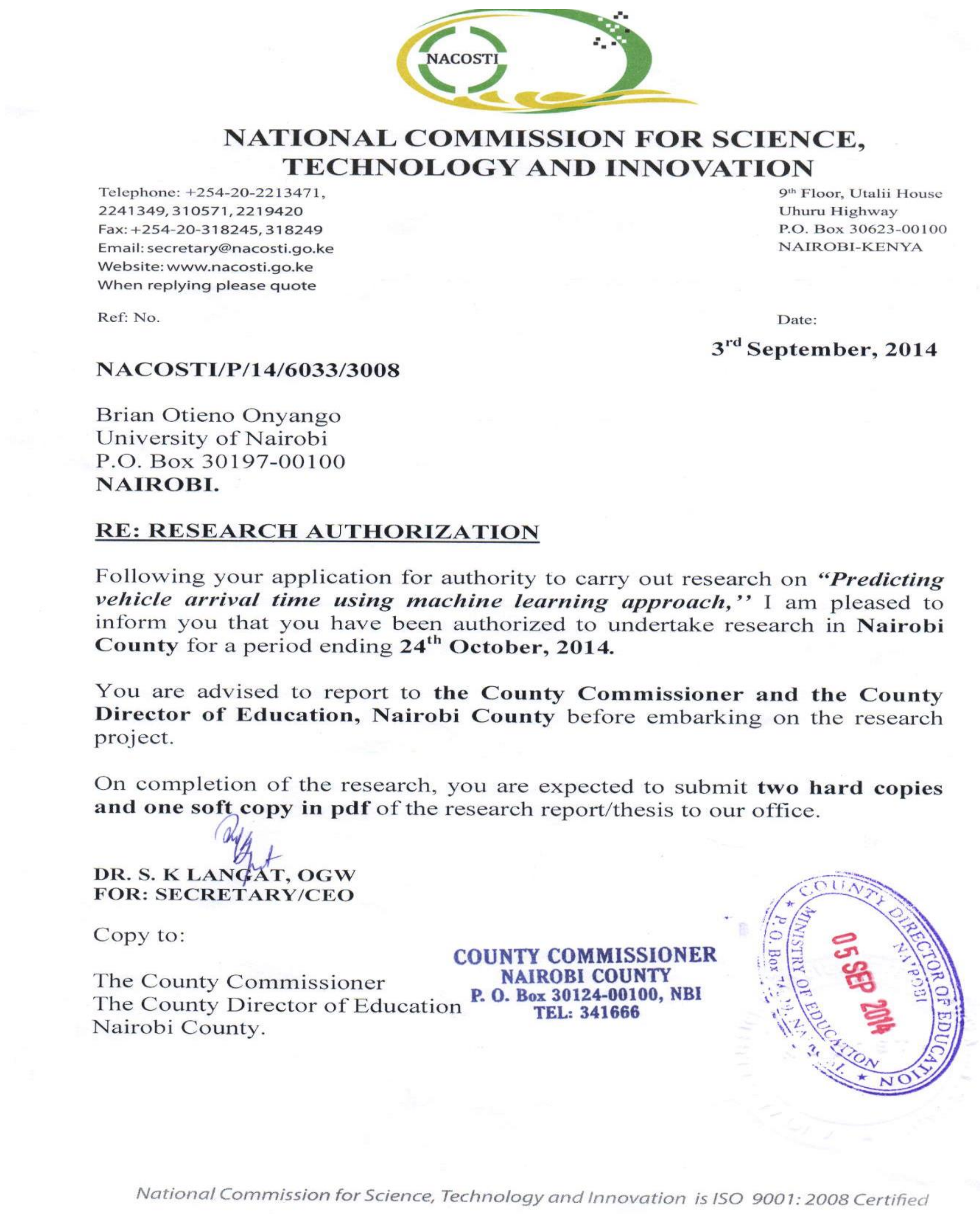
………………………………………………… ……………………………………………….

Witnesses:

…………………………………………………………….

…………………………………………………………….

4. LETTER OF AUTHORIZATION



NACOSTI

NATIONAL COMMISSION FOR SCIENCE, TECHNOLOGY AND INNOVATION

Telephone: +254-20-2213471,
2241349, 310571, 2219420
Fax: +254-20-318245, 318249
Email: secretary@nacosti.go.ke
Website: www.nacosti.go.ke
When replying please quote

9th Floor, Utalii House
Uhuru Highway
P.O. Box 30623-00100
NAIROBI-KENYA

Ref: No.

Date:

3rd September, 2014

NACOSTI/P/14/6033/3008

Brian Otieno Onyango
University of Nairobi
P.O. Box 30197-00100
NAIROBI.

RE: RESEARCH AUTHORIZATION

Following your application for authority to carry out research on ***"Predicting vehicle arrival time using machine learning approach,''*** I am pleased to inform you that you have been authorized to undertake research in **Nairobi County** for a period ending **24th October, 2014.**

You are advised to report to **the County Commissioner and the County Director of Education, Nairobi County** before embarking on the research project.

On completion of the research, you are expected to submit **two hard copies and one soft copy in pdf** of the research report/thesis to our office.

[illegible]

DR. S. K LANGAT, OGW
FOR: SECRETARY/CEO

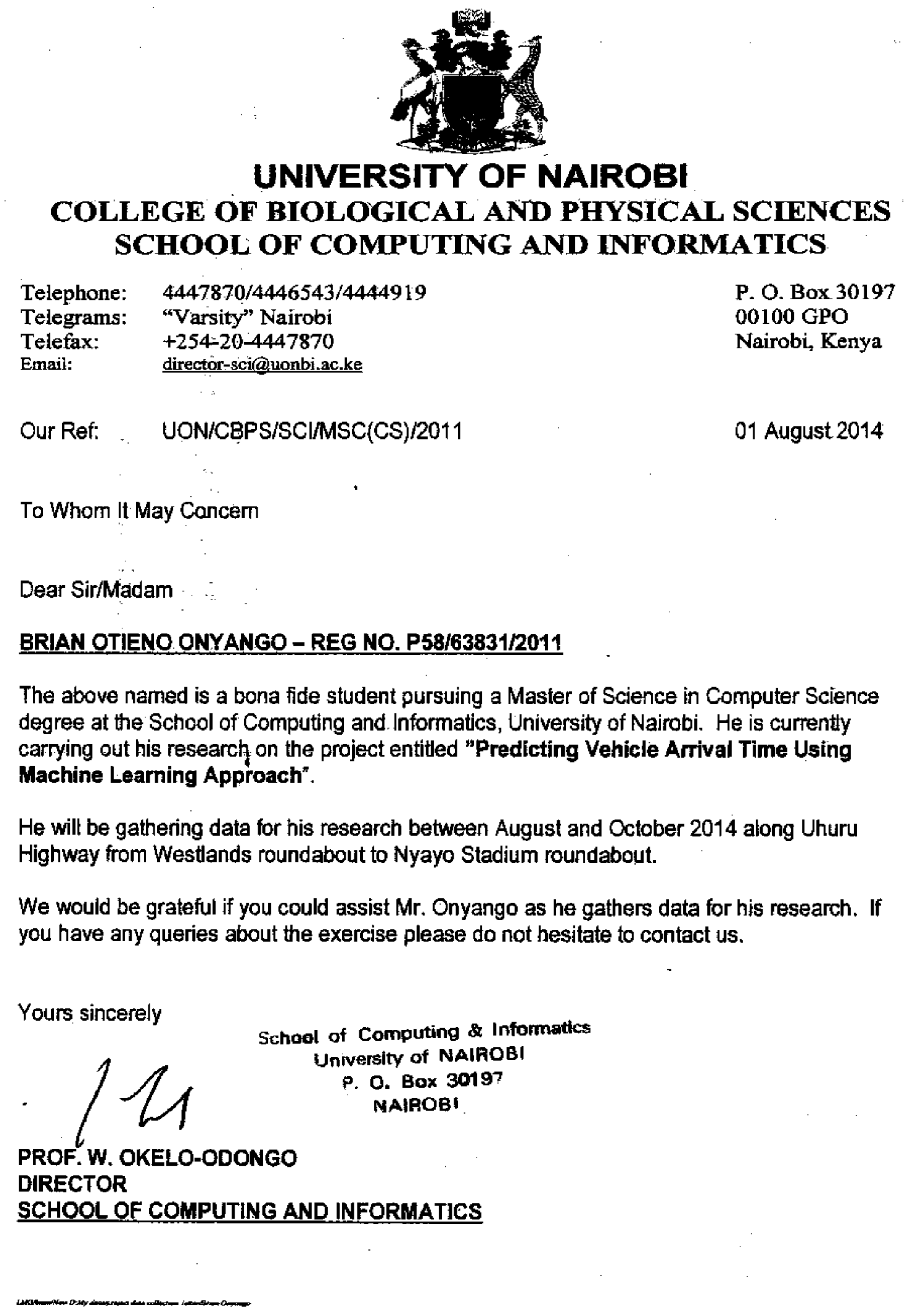
Copy to:

The County Commissioner
The County Director of Education
Nairobi County.

COUNTY COMMISSIONER
NAIROBI COUNTY
P. O. Box 30124-00100, NBI
TEL: 341666

COUNTY DIRECTOR OF EDUCATION
MINISTRY OF EDUCATION
05 SEP 2014
P.O. Box 74... NAIROBI

National Commission for Science, Technology and Innovation is ISO 9001: 2008 Certified



UNIVERSITY OF NAIROBI
COLLEGE OF BIOLOGICAL AND PHYSICAL SCIENCES
SCHOOL OF COMPUTING AND INFORMATICS

Telephone:	4447870/4446543/4444919	P. O. Box 30197
Telegrams:	"Varsity" Nairobi	00100 GPO
Telefax:	+254-20-4447870	Nairobi, Kenya
Email:	director-sci@uonbi.ac.ke	

Our Ref: UON/CBPS/SCI/MSC(CS)/2011 01 August 2014

To Whom It May Concern

Dear Sir/Madam

BRIAN OTIENO ONYANGO – REG NO. P58/63831/2011

The above named is a bona fide student pursuing a Master of Science in Computer Science degree at the School of Computing and Informatics, University of Nairobi. He is currently carrying out his research on the project entitled **"Predicting Vehicle Arrival Time Using Machine Learning Approach"**.

He will be gathering data for his research between August and October 2014 along Uhuru Highway from Westlands roundabout to Nyayo Stadium roundabout.

We would be grateful if you could assist Mr. Onyango as he gathers data for his research. If you have any queries about the exercise please do not hesitate to contact us.

Yours sincerely

School of Computing & Informatics
University of NAIROBI
P. O. Box 30197
NAIROBI

[illegible]

PROF. W. OKELO-ODONGO
DIRECTOR
SCHOOL OF COMPUTING AND INFORMATICS

[illegible]

APPENDIX D: MODEL DESIGN PYTHON CODES

1. SCALING CODES

```
Python 3.2.5 (default, May 15 2013, 23:06:03) (MSC
v.1500 32 bit (Intel)) on win32
Type "copyright", "credits" or "license()" for more
information.
>>> import numpy as np
>>> from sklearn import cross_validation
>>> from sklearn.svm import SVR
>>> import csv
>>> scale=open("scale.csv","r")
>>> data=csv.reader(scale)
>>>data=((eval(row(0)),eval(row(1)),eval(row(2)),eva
l(row(3)),eval(row(4)),eval(row(5)))for row in data)
>>> X=(row(0)for row in data)
>>> Y=(row(1)for row in data)
>>> x = np.array((X))
>>> x.shape
(1, 3786)
>>> X=x.reshape(3786,1)
>>> y = np.array((Y))
>>> Y=y.reshape(3786)
>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.3,ra
ndom_state=0)
>>> clfrbf = SVR(kernel='rbf',
C=10000,gamma=0.001).fit(X_train, y_train)
>>> from time import time
>>> t0 = time()
>>> t0 = time()
>>> t0 = time()
>>> print("done in %0.3fs" % (time() - t0)) done in
17.454s
>>> t0 = time()
>>> clfrbf.score(X_train, y_train)
0.85910077434839727
>>> t0 = time()
>>> clfrbf = SVR(kernel='rbf', C=10000,
gamma=0.001).fit(X_train, y_train)
>>> print("done in %0.3fs" % (time() - t0))
done in 8.566s
>>> clfrbf.score(X_test, y_test)
0.85739491705128357
>>> scale=open("scale1.csv","r")
>>> data=csv.reader(scale)
>>>
data=((eval(row(0)),eval(row(1)),eval(row(2)),eval(ro
w(3)),eval(row(4)),eval(row(5)))for row in data)
```

```
>>> X=(row(0)for row in data)
>>> Y=(row(1)for row in data)
>>> x = np.array((X))
>>> x.shape
(1, 3786)
>>> X=x.reshape(3786,1)
>>> y = np.array((Y))
>>> Y=y.reshape(3786)
>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.3,
random_state=0)
>>> t0 = time()
>>> clfrbf = SVR(kernel='rbf', C=10000,
gamma=0.001).fit(X_train, y_train)
>>> print("done in %0.3fs" % (time() - t0))
done in 8.736s
>>> clfrbf.score(X_train, y_train)
0.99482364476737462
>>> clfrbf.score(X_test, y_test)
0.99405822896236662
>>> scale=open("scale2.csv","r")
>>> data=csv.reader(scale)
>>>data=((eval(row(0)),eval(row(1)),eval(row(2)),eva
l(row(3)),eval(row(4)),eval(row(5)))for row in data)
>>> X=(row(0)for row in data)
>>> Y=(row(1)for row in data)
>>> x = np.array((X))
>>> x.shape
(1, 3786)
>>> X=x.reshape(3786,1)
>>> y = np.array((Y))
>>> Y=y.reshape(3786)
>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.3,
random_state=0)
>>> t0 = time()
>>> clfrbf = SVR(kernel='rbf', C=10000,
gamma=0.001).fit(X_train, y_train)
>>> print("done in %0.3fs" % (time() - t0))
done in 21.352s
>>> clfrbf.score(X_train, y_train)
0.99705345444656546
>>> clfrbf.score(X_test, y_test)
0.99652777459575537
```

2. GRID SEARCH CODES

```
Python 3.2.5 (default, May 15 2013, 23:06:03) (MSC
v.1500 32 bit (Intel)) on win32
Type "copyright", "credits" or "license()" for more
information.
>>> import numpy as np
>>> from sklearn import cross_validation
>>> from sklearn.svm import SVR
>>> import csv
>>> from time import time
>>> from sklearn.grid_search import GridSearchCV
>>> scale=open("scale2.csv","r")
>>> data=csv.reader(scale)
>>>data=((eval(row(0)),eval(row(1)),eval(row(2)),eval(ro
w(3)),eval(row(4)),eval(row(5)))for row in data)
>>> X=(row(0)for row in data)
>>> Y=(row(1)for row in data)
>>> x = np.array((X))
>>> x.shape
(1, 3786)
>>> X=x.reshape(3786,1)
>>> y = np.array((Y))
>>> Y=y.reshape(3786)
>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.3,
random_state=0)
>>> param_grid = ({'C': (1, 10, 100, 1000), 'kernel':
('linear')})
>>> clflin = SVR(kernel='linear')
>>> pos = GridSearchCV(clflin, param_grid)
>>> clf = pos.fit(X_train, y_train)
>>> print(clf.best_estimator_)
SVR(C=1, cache_size=200, coef0=0.0, degree=3,
epsilon=0.1, gamma=0.0,  kernel='linear', max_iter=-1,
probability=False, random_state=None,shrinking=True,
tol=0.001, verbose=False)
>>> clf.score(X_train, y_train)
0.99706608038558286
>>> clf.score(X_test, y_test)
0.99654228462876193
>>> clflin = SVR(kernel='linear', C=.8).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)
0.99706645122036552
>>> clflin = SVR(kernel='linear', C=.2).fit(X_train, y_train)
>>> clflin.score(X_train, y_train)
0.99706707446544662
>>> clflin = SVR(kernel='linear', C=.1).fit(X_train, y_train)
>>> clflin.score(X_train, y_train)
0.99706721945536947
>>> clflin.score(X_test, y_test)
0.99654328540325965
>>> param_grid = ({'C': (1, 10, 100, 1000,10000,100000),
'gamma': (0.001,0.001, 0.0001), 'kernel': ('rbf')})

>>> clfrbf = SVR(kernel='linear')

>>> pos = GridSearchCV(clflin, param_grid)

>>> pose = GridSearchCV(clfrbf, param_grid)

>>> clfrbf = pose.fit(X_train, y_train)

>>> print(clfrbf.best_estimator_)

SVR(C=1000, cache_size=200, coef0=0.0, degree=3,

epsilon=0.1, gamma=0.0001, kernel='rbf', max_iter=-1,

probability=False, random_state=None,shrinking=True,

tol=0.001, verbose=False)

>>> clfrbf.score(X_train, y_train)

0.99706495721692556

>>> clfrbf.score(X_test, y_test)

0.99654005618081176
```

3. TRAINING AND PREDICTION CODES

```
Python 3.2.5 (default, May 15 2013, 23:06:03) (MSC v.1500 32 bit
(Intel)) on win32

Type "copyright", "credits" or "license()" for more information.

>>> import numpy as np

>>> from sklearn import cross_validation

>>> from sklearn.svm import SVR

>>> import csv

>>> sect=open("data.csv","r")

>>> data=csv.reader(sect)

>>>
data=((eval(row(0)),eval(row(1)),eval(row(2)),eval(row(3)),eval(ro
w(4)),eval(row(5)))for row in data)

>>> X=(row(0)for row in data)

>>> Y=(row(1)for row in data)

>>> x = np.array((X))

>>> x.shape

(1, 3785)

>>> X=x.reshape(3785,1)

>>> y = np.array((Y))

>>> Y=y.reshape(3785)

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.3,
random_state=0)

>>> from sklearn import cross_validation as CV

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> from sklearn import metrics

>>> scores.mean()

0.75044634040905867

>>> X=X*100

  >>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.3,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.83206162141786211
```

```
>>> X

array((( 35.9027778),

      ( 36.3194444),

      ( 36.5972222),

      ...,

      ( 70.1388889),

      ( 70.4166667),

      ( 70.6944444)))

>>> Y=Y*100

>>> Y

array(( 36.3194444,  36.875    ,  37.1527778, ...,  70.7638889,

       70.7638889,  71.3888889))

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.3,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99649148989122982

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_test, y_test)

0.99797368106503503

>>> scores

array(( 0.99903225,  0.99623639,  0.99959421,  0.99398558,
0.99899756,

       0.99566191,  0.99750678,  0.99153876,  0.99298413,
0.99937732))

>>> clfrbf.score(X_train, y_train)

0.99647835664067286

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.388146241029

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_test), y_test))

mean squared error: 0.209815692014

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99647961342496949

>>> clflin.score(X_test, y_test)

0.99797487098242799

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.38800772158

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.209692481725

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99649248106167521

>>> clfreg=LinearRegression()

Traceback (most recent call last):

  File "<pyshell#47>", line 1, in <module>

    clfreg=LinearRegression()

NameError: name 'LinearRegression' is not defined

>>> from sklearn.linear_model import LinearRegression

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()
```

```
0.99650854709401904

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99649966835638837

>>> clfreg.score(X_test, y_test)

0.99797619123016623

>>> print ("mean squared error:", metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.385797319943

>>> print ("mean squared error:", metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.209555776349

>>> X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,Y,test_size=0.2, random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10, scoring='r2')

>>> scores.mean()

0.99685914020555022

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train, y_train)

>>> clfrbf.score(X_train, y_train)

0.99678550158204249

>>> clfrbf.score(X_test, y_test)

0.9974222834241705

>>> print ("mean squared error:", metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.351452423076

>>> print ("mean squared error:", metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.267230526202

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99678652098753417

>>> clflin.score(X_test, y_test)

0.99742376132900479

>>> print ("mean squared error:", metrics.mean_squared_error(clfrbf.predict(X_test), y_test))

mean squared error: 0.267420662799

>>> X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,Y,test_size=0.1, random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10, scoring='r2')

>>> scores.mean()

0.99663697841921939

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train, y_train)

>>> clfrbf.score(X_train, y_train)

0.99660638434599891

>>> clfrbf.score(X_test, y_test)

0.99973766057473556

>>> print ("mean squared error:", metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.368887653413

>>> print ("mean squared error:", metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.0273176092489

>>> X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,Y,test_size=0.05, random_state=0)
>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99677250445281873

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_train, y_train)

0.99675625690980596

>>> clfrbf.score(X_test, y_test)

0.99972107948245637

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.3511258993

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.0298616530557

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.05,
random_state=0)

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.4,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99640433644508097

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_train, y_train)

0.99632636865629653

>>> clfrbf.score(X_test, y_test)

0.99783735742483237

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.407563310179

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.224999488224

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.5,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99590780940004764

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_train, y_train)

0.99581572847723043

>>> clfrbf.score(X_test, y_test)

0.99805866291777012

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.464583077091

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.204590568776

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.4,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)
>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99640433644508097

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_train, y_train)

0.99632636865629653

>>> clfrbf.score(X_test, y_test)

0.99783735742483237

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.407563310179

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.224999488224

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.3,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.4,
random_state=0)

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.6,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99584768445645566

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_train, y_train)

0.99582737940735755

>>> clfrbf.score(X_test, y_test)

0.99763770533076312

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.4546681957

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.254435546907

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.7,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99478388869412238

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_train, y_train)

0.99490088736870341

>>> clfrbf.score(X_test, y_test)

0.99780727127306679

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.568976247495

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.23409196611
```

```
>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.8,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99456518080850242

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_train, y_train)

0.99444151365022015

>>> clfrbf.score(X_test, y_test)

0.99755321424414245

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.623763788832

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.262178397052

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.9,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99942942837141335

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_train, y_train)

0.99943762713482165

>>> clfrbf.score(X_test, y_test)

0.99658251802845566

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.063249567319

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.364724920397

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.02,
random_state=0)

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001)

>>> scores = CV.cross_val_score(clfrbf,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99688098197064201

>>> clfrbf = SVR(kernel='rbf', C=1000, gamma=0.0001).fit(X_train,
y_train)

>>> clfrbf.score(X_train, y_train)

0.9968393600896327

>>> clfrbf.score(X_test, y_test)

0.99973409928752233

>>> print ("mean squared error:",
metrics.mean_squared_error(clfrbf.predict(X_train), y_train))

mean squared error: 0.341320519091

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.0304880062944

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.4,
random_state=0)

>>> clfreg=LinearRegression()
>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10, scoring='r2')

>>> scores.mean()

0.99641811091997445

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99634904488654275

>>> clfreg.score(X_test, y_test)

0.99783963612932236

>>> print ("mean squared error:", metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.405047543463

>>> print ("mean squared error:", metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.224819670437

>>> X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,Y,test_size=0.5, random_state=0)

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10, scoring='r2')

>>> scores.mean()

0.99592029531815829

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99584104568291942

>>> clfreg.score(X_test, y_test)

0.99806125963032288

>>> print ("mean squared error:", metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.461772087112

>>> print ("mean squared error:", metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.204345271041

>>> X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,Y,test_size=0.6, random_state=0)

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10, scoring='r2')

>>> scores.mean()

0.99586424721161126

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99585757548323506

>>> clfreg.score(X_test, y_test)

0.99763891627045942

>>> print ("mean squared error:", metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.451377890475

>>> print ("mean squared error:", metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.254327246931

>>> X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,Y,test_size=0.7, random_state=0)

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10, scoring='r2')

>>> scores.mean()

0.99480119005833667

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)
```

```
0.99494066179752561

>>> clfreg.score(X_test, y_test)

0.99780196048391878

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.564538082093

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.234639512941

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.8,
random_state=0)

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99458859242566822

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99449751508006001

>>> clfreg.score(X_test, y_test)

0.99753832545829746

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.617479404584

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.263828920189

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.9,
random_state=0)

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99944503228852177

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99944665121021425

>>> clfreg.score(X_test, y_test)

0.99661382042983637

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.0622346377245

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.364840664337

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.2,
random_state=0)

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99687198145897837

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99680371289560321

>>> clfreg.score(X_test, y_test)

0.99743143063047357

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.349461316083
```

```
>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.266471702004

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.1,
random_state=0)

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99665244159086974

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99662657030015944

>>> clfreg.score(X_test, y_test)

0.9997214472181114

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.366693430489

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.0289249245271

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.05,
random_state=0)

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99679065644910292

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99677822088209456

>>> clfreg.score(X_test, y_test)

0.99970086456200447

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.348748362205

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.032199907905

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.02,
random_state=0)

>>> clfreg=LinearRegression()

>>> scores = CV.cross_val_score(clfreg,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99689915967086318

>>> clfreg=LinearRegression().fit(X_train, y_train)

>>> clfreg.score(X_train, y_train)

0.99686065073269292

>>> clfreg.score(X_test, y_test)

0.99973277894392132

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_train), y_train))

mean squared error: 0.339021322236

>>> print ("mean squared error:",
metrics.mean_squared_error(clfreg.predict(X_test), y_test))

mean squared error: 0.0317901269517

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')
```

```
>>> scores.mean()

0.99688631387257998

>>> clflin.score(X_train, y_train)

0.99684412784350518

>>> clflin.score(X_test, y_test)

0.99974496963608861

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.340805644802

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.0303398533194

>>>

X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.05,
random_state=0)

>>> clflin = SVR(kernel='linear', C=1)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99677681441522137

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99676105430833151

>>> clflin.score(X_test, y_test)

0.99972311576017092

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.350606594649

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.0298047168285

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.1,
random_state=0)

>>> clflin = SVR(kernel='linear', C=1)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99663807310914654

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99660764296540794

>>> clflin.score(X_test, y_test)

0.99973738255311839

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.368750840878

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.0272701991308

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.2,
random_state=0)

>>> clflin = SVR(kernel='linear', C=1)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99686000900555105

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99678652098753417
```

```
>>> clflin.score(X_test, y_test)

0.99742376132900479

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.351340967886

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.267267340167

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.4,
random_state=0)

>>> clflin = SVR(kernel='linear', C=1)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99640570041445553

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99632749172946777

>>> clflin.score(X_test, y_test)

0.99783808203076518

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.407438713186

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.224981398714

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.5,
random_state=0)

>>> clflin = SVR(kernel='linear', C=1)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99590942431839657

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99581771103939154

>>> clflin.score(X_test, y_test)

0.99805942605049514

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.464362951599

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.204538532281

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.6,
random_state=0)

>>> clflin = SVR(kernel='linear', C=1)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99584937533187623

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99582895270719851

>>> clflin.score(X_test, y_test)

0.9976384832827887

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.454496761613
```

```
>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.254373886769

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.7,
random_state=0)

>>> clflin = SVR(kernel='linear', C=1)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99478636688902533

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99490365563771033

>>> clflin.score(X_test, y_test)

0.99780826940183565

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.568667354668

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.233966039413

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.8,
random_state=0)

>>> clflin = SVR(kernel='linear', C=1)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.9945676088075972

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99444469222079535

>>> clflin.score(X_test, y_test)

0.99755447445209167

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.623407095103

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.262098158659

>>> X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X,Y,test_size=0.9,
random_state=0)

>>> clflin = SVR(kernel='linear', C=1)

>>> scores = CV.cross_val_score(clflin,X_train, y_train, cv=10,
scoring='r2')

>>> scores.mean()

0.99944270779349209

>>> clflin = SVR(kernel='linear', C=1).fit(X_train, y_train)

>>> clflin.score(X_train, y_train)

0.99944452013751062

>>> clflin.score(X_test, y_test)

0.99660079174150118

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_train), y_train))

mean squared error: 0.0624743175433

>>> print ("mean squared error:",
metrics.mean_squared_error(clflin.predict(X_test), y_test))

mean squared error: 0.366244427844

>>>
```

APPENDIX E: PROTOTYPE CODES

1. USER INTERFACE

```
<html>
  <head>
    <meta name="viewport"
        content="initial-scale=1.0, user-scalable=no" />
    <style type="text/css">
      html { height: 100% }
      body {  height: 80%; margin: 10; padding: 10 }
      #map-table { position:absolute;
bottom:45%;left:15%;border:2px;}
    </style>
</head>
<body bgcolor="#E6E6FA";>
<form id="map-table" name="input" action="map.php"
method="get">
<table  align="center" valign="center">
<td><select  name="address1" id="address1" >
<option selected="selected">-Select Location-</option>
<?php
include('db.php');
$sql=mysql_query("select id, name from road");
while($row=mysql_fetch_array($sql))
{
$id=$row('id');
$data=$row('name');
echo '<option value="'.$id.'">'.$data.'</option>';
 } ?>
</select></td>
<td><select  name="address2"  id="address2" ;>
<option selected="selected">-Select Location-</option>

<?php
include('db.php');
$sql=mysql_query("select id, name from road");
while($row=mysql_fetch_array($sql))
{
$id=$row('id');

$data=$row('name');
echo '<option value="'.$id.'">'.$data.'</option>';
 } ?>
</select></td>
</table>
</form>
</body>
</html>
```

2. QUERY CODES

```
<?php
  include 'connect.php';
  $apikey =
"AIzaSyDnwJWOqnkS3yIANAt_AN2ZzDmE14PxtzE";
  $sid = $_REQUEST('address1');
$fid =  $_REQUEST('address2');
$time=$_REQUEST('time');
$etime=strtotime($time);
$current=date("h:i A" ,time());
$stime=date("h:i A",$etime );
$findmap = "SELECT latitude, longitude FROM road
WHERE ID = '$sid'";
   if(!$result = $con->query($findmap)){
    die('There was an error running the query (' . $con-
>error . ')');
  } else {
   $row = $result->fetch_assoc();
   $lat = $row('latitude');
   $long = $row('longitude');
   $zoom = 10;
$findmape = "SELECT latitude, longitude FROM road
WHERE ID = '$fid'";
if(!$result2 = $con->query($findmape)){
    die('There was an error running the query (' . $con-
>error . ')');
  } else {
   $row2 = $result2->fetch_assoc();
   $lat2 = $row2('latitude');
   $long2 = $row2('longitude');
$cenlat=($lat+$lat2)/2;
$cenlog=($long+$long2)/2;
  }
  }
 ?>
<!DOCTYPE html>
<html>
  <head>
   <meta name="viewport"
     content="initial-scale=1.0, user-scalable=no" />
   <style type="text/css">
    html { height: 100% }
    body {  height: 80%; margin: 10; padding: 10 }
    #map-canvas { position:absolute;
bottom:5%;right:5%;border:2px;}
.   #map-table { position:absolute; bottom:30%;left:5%;border:2px;}
   </style>
   <script type="text/javascript"
     src="https://maps.googleapis.com/maps/api/js?key=
       <?php echo $apikey; ?>&sensor=false">
   </script>
   <script type="text/javascript">
     function initialize() {
      var center = new google.maps.LatLng(<?php echo
$lat.', '.$long; ?>);
      var center2 = new google.maps.LatLng(<?php echo
$lat2.', '.$long2; ?>);
  center: latlng,
        zoom: <?php echo $zoom; ?>       };
      var map = new
google.maps.Map(document.getElementById("map-
canvas"),
         mapOptions);
marker = new google.maps.Marker({
            map: map,
            position: center,
         });
var marker = new google.maps.Marker({
            map: map,
            position: center2,
     }
     google.maps.event.addDomListener(window, 'load',
initialize);
   </script>
  </head>
  <body bgcolor="#E6E6FA";>
 <div id="headpos" style="width:100%; height:5%"
vertical-align: top;><h1 align="center">ESTIMATED
ARRIVAL TIME</h1> </div>
 <div id="map-table" style="width:50%;>
 <table border="1" align="left">
 <tr>
<td >
<h4>Current Time: <?php echo $current ?></h4>
<h4>Start Time: <?php echo $stime ?></h4>
<h4><?php
for($x=$sid;$x<$fid;$x++)
{
include('db.php');
switch ($x) {
  case 1:
   $sql=mysql_query("select * from secone where
timein='$time'");
while($row2=mysql_fetch_array($sql)){
    $id2 = $row2('id');
    $time2 = $row2('timeout');
echo "<table width='350'>";
echo "<tr>";
echo "<td> Bunyala Roundabout: </td>";
echo "<td>" . $time2 . "</td>";
$time=$time2;
echo "</tr>";
echo "</table>";}
    break;
  case 2:
    $sql=mysql_query("select * from sectwo where
timein='$time'");
while($row2=mysql_fetch_array($sql)){
    $id2 = $row2('id');
    $time2 = $row2('timeout');
echo "<table  width='350'>";
echo "<tr>";
echo "<td> Haile Selassie Roundabout </td>";
echo "<td>" . $time2 . "</td>";
$time=$time2;
echo "</tr>";
echo "</table>";}
    break;
  case 3:
    $sql=mysql_query("select * from secthree where
timein='$time'");
while($row2=mysql_fetch_array($sql)){
    $id2 = $row2('id');
    $time2 = $row2('timeout');
echo "<table  width='350'>";
echo "<tr>";
echo "<td> Kenyatta Avenue Roundabout </td>";
echo "<td>" . $time2 . "</td>";
$time=$time2;
echo "</tr>";
echo "</table>";}
    break;
  case 4:
    $sql=mysql_query("select * from secfour where
timein='$time'");
while($row2=mysql_fetch_array($sql)){
    $id2 = $row2('id');
    $time2 = $row2('timeout');
echo "<table  width='350'>";
echo "<tr>";
echo "<td> University way Roundabout</td>";
echo "<td>" . $time2 . "</td>";
$time=$time2;
echo "</tr>";
echo "</table>";}
    break;
  case 5:
    $sql=mysql_query("select * from secfive where
timein='$time'");
while($row2=mysql_fetch_array($sql)){
    $id2 = $row2('id');
    $time2 = $row2('timeout');
echo "<table  width='350'>";
echo "<tr>";
echo "<td> Westlands Roundabout</td>";
echo "<td>" . $time2 . "</td>";
$time=$time2;
echo "</tr>";
echo "</table>";}}
$time_diff = (strtotime($time) - strtotime($stime));
$mins=($time_diff)/60;
?>
</table>
</h4>
<h4>Time taken: <?php echo $mins ?> Minutes</h4>
</td>
</tr>
 </table>
  </div>
<div id="map-canvas" style="width:50%; height:80%"
vertical-align: bottom;/>
  </body>
</html>
```