

CROWD SOURCE BASED TRAFFIC ANALYSIS USING MACHINE LEARNING ALGORITHM

By

Marzia Khan Orthy	ID: 12101049
Suraiya Sharmin	ID: 12101057
Ashraful Islam Reshad	ID: 12101059



Inspiring Excellence

**Department of Computer Science Engineering
BRAC University**

Supervised by: Dr. Amitabha Chakrabarty

This report submitted to the BRAC University in accordance with the requirements of the degree of BACHELOR IN COMPUTER SCIENCE AND ENGINEERING in the Dept. of Engineering & Computer Science.

Submitted on: 21st August, 2017

Author's Declaration

Thesis Submission to the Department of Computer Science and Engineering, BRAC University, Dhaka, submitted by the authors for the purpose of obtaining the degree of Bachelor of Science in Computer Science. We hereby announce that the results of this thesis are entirely based on our research. Resources taken from any research conducted by other researchers are mentioned through reference. This thesis either in whole or in part, has not been previously submitted for any degree.

Signature of Supervisor:

Signature of Authors:

Dr. Amitabha Chakrabarty

Assistant Professor

Department of Computer Science and

Engineering

BRAC University

1. Marzia Khan Orthy (ID: 12101049)

2. Suraiya Sharmin (ID: 12101057)

3. Ashraful Islam Reshad (ID: 12101059)

Acknowledgement

The majority of the credit for this thesis goes to our honorable supervisor, Dr. Amitabha Chakrabarty. This concept was his brainchild and he has been a constant beacon of guidance and encouragement throughout this whole task. His support has enabled us to go against the odds and complete this thesis. We express our heartfelt gratitude towards his great efforts and sincere well-wishes behind our endeavor.

We would like to thank our friend and family who supported us wholeheartedly throughout the completion of the thesis. Special thanks goes to Nowfel Mashnoor, Reezwan Ahmed & Ponir Hossain, who provided us valuable suggestions and guidance in many stages of this work. We also thank our family who supported us financially and mentally even when we were under severe stress. Without them, the work would not have seen the face of completion.

We thank the numerous contributors whose commute data has helped enrich the database. They have gone into great trouble into meticulously recording their journey time and also painstakingly providing the inputs. We are grateful for the time and effort they have invested into this work. We would also like to thank in advance those who would continue to contribute to our crowdsourced project. It is our sincere hope that the outcome of the project will be of use to them and others.

Abstract

One of the most detrimental effects to our economy currently is most certainly Traffic jam. Be it in a public vehicle or private, valuable work hours (around 3.2 million per day) is wasted everyday while waiting in the traffic. While this problem cannot be overcome without proper urban planning and traffic management, there are definitely ways of providing the commuters with an idea about how long they might be needing for their route. Often, this information might decide between rescheduling a meeting or missing it altogether. Keeping this in mind, the Dhaka Real Traffic project has been taken. It aims to provide travel time predictions based on machine learning of crowd sourced real commuting data.

Data mining was done by means of a data collection app and also via Google form. The collected data was classed and trained by means of Python coding. From an initial choice between SVM, KNN & ANN, ANN was selected as the machine learning algorithm due to its lowest mean square errors among all three. Using Java and XML, the frontend Android App name Dhaka Real Traffic (DRT) was created with backend server learning. Due to machine learning, DRT will continue to upgrade its database to provide the most realistic travel time estimate.

Table of Contents

List of Figures.....	vii
Chapter 1.0 Introduction	1
1.1 Objectives	2
1.1.1 Primary Objective	2
1.1.2 Secondary objectives.....	2
1.2 Rationale.....	2
1.3 Scope	3
1.4 Limitations	4
1.5 Methodology	5
1.5.1 Data collection.....	5
1.5.2 Data Analysis	5
1.5.3 Implementation.....	5
1.6 Work Process	6
1.6.1 Overall	6
Chapter 2.0 Literature Review	7
2.1 Previous Studies.....	7
2.2 Crowdsourcing	8
2.3 Linear Regression	9
2.4 K-Nearest Neighbor Algorithm (K-NNR Algorithm).....	10
2.5 Decision Tree	12
2.6 Artificial Neural Network	12
2.7 Support Vector Machines (SVM)	13
2.8 Phantom Traffic Factor Prediction.....	14
2.9 FLASK (Webhook/Web Framework).....	15
2.10 Competitor Analysis	15
2.10.1 Google Maps	16
2.10.2 Go! Traffic.....	16
2.10.3 RastaRObosta	17

Chapter 3.0 System Analysis.....	18
3.1 Context Diagram	18
3.2 Decision Tree	18
3.2.1 Contribution	19
3.2.2 User Query	19
3.3 Data Flow	20
3.4 Error Comparison	21
3.5 ANN Algorithm.....	22
3.6 Result Analysis for Error Comparison	24
Chapter 4.0 System Design	27
4.1 Activity Flow	27
4.2 Feasibility study	28
4.2.1 Technical Viability.....	28
4.2.2 Financial Feasibility	28
4.2.3 Legal Feasibility	29
4.3 Design & Architecture	29
4.3.1 Pseudo Codes	29
4.3.1.1 Data Collection	29
4.3.1.2 Backend Data Classing.....	29
4.3.1.3 Data Training	30
4.3.1.4 Webhook (App Implementation)	30
4.3.2 Algorithm.....	30
4.3.2.1 Data Processor	30
4.3.2.2 Data Training	32
4.3.2.3 Webhook.....	32
Chapter 5.0 Implementation	34
5.1 Data Collection App	34
5.2 Mobile Application (DRT)	36
5.2.1 Backend	36
5.2.1.1 Backend (Server) specifics.....	36
5.2.1.2 Programming.....	36
5.2.2 Frontend (Mobile App).....	39

5.2.2.1 Frontend specifics	39
5.2.3 The App in Action	39
5.2.3.1 Interface	40
5.2.3.2 Outputs	41
5.2.3.4 Error validation	42
5.2.3.5 Feedback	43
5.2.3.6 Contribution	44
Chapter 6.0 Conclusion	45
6.1 Conclusion	45
6.2 Future planned developments	46
Bibliography	47

List of Figures

Fig 1.1: Work Process for DRT (Flow)	6
Fig 1.2: Process Cycle for DRT	6
Fig 2.1: Linear Regression.....	9
Fig 2.2: KNN Functions	11
Fig 3.1: Context Diagram	18
Fig 3.2: Decision Tree for User Contribution	19
Fig 3.3: Decision Tree for User Query	19
Fig 3.4: Data Flow Diagram	20
Fig 3.5: Error comparison between machine learning languages	21
Fig 3.6: ANN error profile showing descent gradient	21
Fig 3.7: Training process with descent gradient	23
Fig 3.8: Descent Gradient	23
Fig 3.9: Error Comparison between 3 analytical models in Abdullahpur-Agargaon route.....	24
Fig 3.10: Error Comparison between 3 analytical models in Abdullahpur-Airport route.....	25
Fig 3.11: Error Comparison between 3 analytical models in Abdullahpur-Azimpur route	25
Fig 3.12: Error Comparison between 3 analytical models in Abdullahpur-Badda route	26
Fig 5.1: Data Collection Entry Screen	35
Fig 5.2: Data Collection Completed Screen	35
Fig 5.3: Data Collection Auto Generated Summary/Analysis	35
Fig 5.4: Data Collected in free hosted website (Blackjack).....	36
Fig 5.5: Appending Data	37
Fig 5.6: Data mapping for Training	37
Fig 5.7: Mapped Location Data	38
Fig 5.8: Data Training	38
Fig 5.9: Webhook Coding.....	39
Fig 5.10: Open Screen	40
Fig 5.11: Completed Query Information	40
Fig 5.12: Output 1	41
Fig 5.13: Output 2	41
Fig 5.14: Output 3	41
Fig 5.15: Output 4.....	41
Fig 5.16: Location Error Validation	42
Fig 5.17: Time Error Validation	42
Fig 5.18: Feedback Entry.....	43
Fig 5.19: Feedback Acceptance	43
Fig 5.20: Contribution Entry.....	44
Fig 5.21: Contribution Acceptance	44

Chapter 1.0 Introduction

When asked about the main problem in a Dhaka dweller's life, one of the most frequent answers would be 'stuck in traffic'. Traffic congestion eats up valuable personal and professional time, with detrimental effects to social, professional, economic and overall productive activities. Proper time planning and scheduling is a crucial issue for both personal and professional purposes. Lack of proper planning often causes whole day to be wasted without any proper progress. So, in order to come up with a solution to reduce the traffic congestion issue of Dhaka city, the 'Dhaka Real Traffic' (DRT), a Crowd Source Based Traffic Analysis app development was initiated.

It will forecast and provide a time range within which a commuter can expect to make the journey. However, it must be noted that the travel time is not constant. While there are many apps that provide a map and travel time between two places in different transports (e.g. Google Maps), they often consider an approximate speed and calculate using that value as constant. However, the traffic scenario of our country, especially Dhaka city should be considered. The traffic scenario varies from day to day or even time of the day. The traffic congestion during rush hours is quite incomparable to that of the off-peak hours. Also, the traffic during the 1st and last day of the week may vary drastically on particular routes, such as those leaving or entering the city as there is a greater tendency for people to leave Dhaka before the weekend or enter after the weekend. Hence the human behavior plays a major role in the traffic density of the roads. With careful and detailed analysis, a trend may be established in the travel pattern and thereby the traffic density pattern.

Another factor affecting the travel pattern and traffic situation is phantom traffic. This is where a bottleneck is created in the road without any apparent reason. This is because as one person brakes for any reason, the one behind has to brake automatically and so on. This creates an automatic slowdown and subsequent stagnant traffic in the road. Often, there is no particular or appropriate cause for this congestion. This situation is known as phantom traffic. Therefore, we would be using phantom traffic prediction and equation to try to identify the amount of phantom traffic that may be experienced in a particular route.

SVM (Support Vector Machine) algorithm will be used for supervised machine learning. It will incorporate past data and regularly updated data alongwith informative dummy data to continually update its results.

Other forms of analysis to be used are regression analysis, factor analysis, K Nearest Neighbor Regression (KNNR), etc. Regression analysis would allow us to find the relation between route and time taken and also between time of the day and time taken. Factor analysis would enable us to identify the level of impact of each of the parameters that affect the traffic situation. All of these analyses combined would likely produce a sufficiently accurate prediction for the time required for the journey.

Apart from the technical aptitude, the app would also have a well-developed GUI that is user-friendly and easy to use. There would be integrity within the different forms. For the report, the theoretical aspects have been covered from past secondary sources. The data collection is done via a data collection app.

1.1 Objectives

The outcome of the thesis is to meet the following objectives:

1.1.1 Primary Objective

To successfully correlate the information from data collected to create an accurate model for travel time prediction within Dhaka city.

1.1.2 Secondary objectives

- Daily travel time data collection from commuters over a period of time.
- Analysis of collected data into multiple models.
- Trial of the different models to properly identify and improve the best model.
- Implement the design into an android app for general use.

1.2 Rationale

Recently a World Bank report has found out that the average traffic speed of Dhaka has fallen from 21 kmph to 7 kmph. This is slightly more than the average human walking speed. Moreover,

traffic congestion causes a loss of 3.2 million working hour every day. One of the most detrimental facts for hamper of economic growth can easily be identified as traffic congestion. Without any proper guidelines, commuters cannot properly plan their schedules and, therefore, often miss out on important appointments or meetings.

The critical need for a solution to this problem has served as a rationale for development of DRT. We have tried to provide a solution that can allow commuters to properly plan their schedules. As DRT provides reflection of real traffic congestion possibilities, people can schedule their activities on times with lower commute times. We hope to contribute to the aggregate efficiency increase of Dhaka dwellers.

DRT will provide realistic travel time estimate when user enters start & end points and mode of transport. It automatically integrates the date and time to fit it into the regression analysis from past data and project a realistic travel time. Since past data has been taken over a period of time, it takes into account the various random factors such as phantom traffic, VIP movement, road works, vehicle availability, etc. Due to incorporation of ANN, DRT will continue to enrich its regressions to provide even more accurate predictions as use increases.

The rationale behind implementing DRT as android app is that the mobile platform has become the most accessible and available and DRT needs to be used on-the-go. This will enable us to keep DRT in a lite version while serving the purpose to reach maximum populace in the best possible way.

The rationale behind choosing Dhaka as the subject is very obvious. Traffic is not a problem in other parts of the country as much as it is in Dhaka. Dhaka citizen will be needing it the most. Later plan include extension to Chittagong as well. But, apart from that further expansion as per current scenario will not be a viable option.

1.3 Scope

Though our ultimate intention maybe a globally applicable platform, for our current project, we have limited scope as follows:

- **Region:** The area covered is within Dhaka city. Moreover, DRT covers transit between major stoppages and not any point the map.

- **Range of vehicles:** DRT uses private car and buses. However, these are broad categories. Private car includes all sort of private cars other than motorbikes and may also include CNG driven autorickshaws. Bus also includes other form of public transport like leguna as well.
- **Time range:** As DRT uses actual crowds sourced data, the data accuracy will be comparatively high for typical waking time of citizens (6am-12am).

1.4 Limitations

As aspiring CSE professionals, we are still in our student stages and have faced some limitations, like resource, experience, etc. during completion of this thesis. Some are as follows:

- **Data gathering volume:** Even though data gathering was carried out over more than 6 months, it was not possible to get regular data on travel time between all the nodes. Also, many of the data submitted was during office/school/university commute time. Moreover, there was very less data during late night commute time. As a result, the analysis has a possibility of being skewed.
- **Lack of resources:** As it is crowd source based app, a huge number of applicant is necessary. In order to spread the news and convince people to submit data regularly, we needed to invest a large amount of time daily, which could not be properly given due to other courses to be completed as well.
- **Insufficient dry run:** The final outcome is created on theoretical models and analysis. How accurate it will work can be properly understood over a period of time, when it can be identified if its SVM is working accurately as well.
- **Platform:** Currently, DRT will only be available in Android platform. This is because we have most expertise here and also Android is the most widely used platform. However, this will not be available at iOS or Windows or PC for the time being.

1.5 Methodology

The whole thesis project can be separated into 3 major segments: data collection, analysis and implementation. All of these were very meticulously handled in depth. As DRT is crowd sourced, we had to focus initially on the crowd. Let us look into the methodology in detail.

1.5.1 Data collection

The final app is a very data extensive app and requires massive and continuous regular input of actual travel data from commuters. Data collection was collected on multiple fronts. As there are many types of users, we kept multiple options open for data entry. If anyone wished, he/she could keep manual entry in an excel sheet. Also, a Google doc was created and the link shared for data entry. Finally, a data collection app was created for regular updates. Later, data from all these were compiled into a single database. However, it was later decided that only the app would be used for the data collection as this approach is similar to the final app that would be used and commuters would get a real feel and get used in advance. Also, as it can be carried in smartphones, it increases the level of ease for the data entry.

1.5.2 Data Analysis

After data collection, they will be analyzed by different methods. The methods to be used are KNNR, Linear Regression, decision tree, Neural Network, SVM, etc. Final selection will be made considering lowest error among all the analyses.

1.5.3 Implementation

Final implementation will be via a mobile app that takes start and end point as input data only. It will collect start time and date from mobile system and provide approximate end time by calculation as per analysis from existing data. Moreover, there will be options for feedback and contribution. Contribution will allow a user to enter their own travelling data, which in turn will upgrade the server database. Feedback will allow them to comment their experience or suggestion.

1.6 Work Process

The following shows the general overall procedure for the gross activities for DRT in both flow and circular format.

1.6.1 Overall

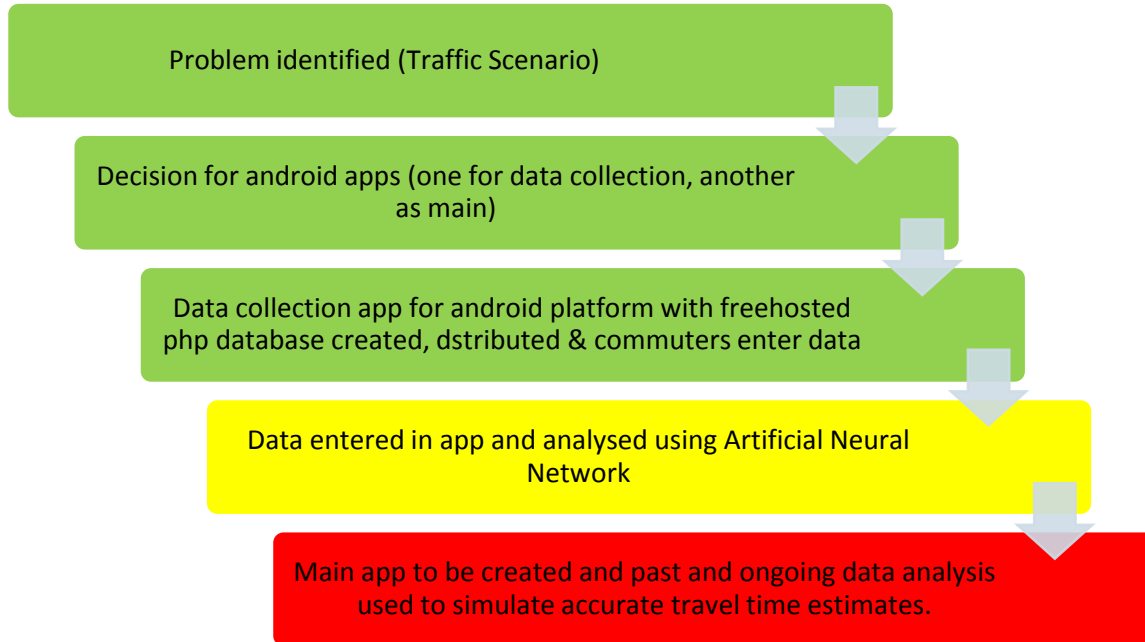


Fig 1.1: Work Process for DRT (Flow)

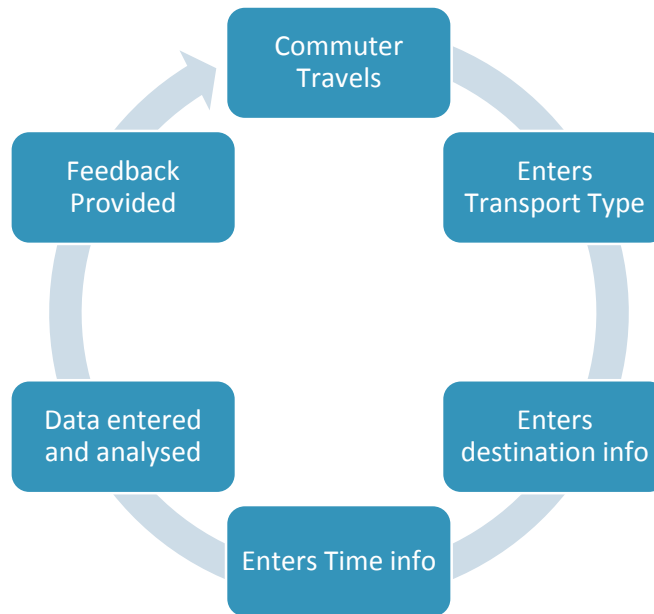


Fig 1.2: Process Cycle for DRT

Chapter 2.0 Literature Review

For every work, it is important to recognize the benefit of secondary sources. It is always valuable to tap into the existing similar works done by scholars and researchers. Similarly, it is also crucial to have a widespread knowledge regarding the major points which are needed for the completion of the report. This section focuses on the existing materials.

2.1 Previous Studies

Many studies have gone to assess the traffic scenario and identify the cause root for it. So far, many studies have come forward with many underlying factors. These past analyses have provided us with sufficient rationale to identify this issue as one of the most critical in the country.

Recently, World Bank has identified average speed of Dhaka Traffic to have fallen from 21kmph to 7kmph (a little more than average walking speed of 5kmph). Moreover 3.2 million work hours are wasted everyday due to this congestion (Source: all leading newspaper of Bangladesh).

Past studies have been conducted in other countries regarding travel time prediction and traffic flow as well. A study on time series prediction has taken historical real time traffic flow data over a year and used *feed forward neural networking* to predict the future traffic flow. According to the study, Prediction of traffic can be used to improve the traffic condition and reduces travel time having the available capacity. Prediction system uses the emerging computer, communication and control technologies for managing and monitoring the transportation. Many factors such as weather condition, exhibitions and holidays can affect the quality of the traffic forecasting. One of the prediction methods is the time-series forecasting. In time-series forecasting, the historical data are collected and analyzed to make a model. Then this model is extrapolated for forecasting the future values (Raeesi, Mesgari, Mahmoudi, 2014).

Another study labelled ‘Short-term traffic flow prediction using seasonal ARIMA model with limited input data’ was conducted on a road in Chennai using data from only 3 days. It used the SARIMA modelling and validated prediction for the next 24 hours. Taking the historical and peak values, it attempted to forecast traffic flow and, thereby, the travel time (Kumar, 2015).

A BUET research was also conducted in the 90s labelled 'Development of transport models to predict flow-pattern and volume of traffic on the national highway network'. This serves as a feasibility model an expansion guideline for future infrastructural developments.

All or most of these studies were mainly focused on development of Intelligent Transport System.

However, none of these are serves to provide information to the masses. DRT focuses on the prediction and providing of these information to everyone via android app. Therefore, the analytical models thought to be most appropriate for this project were to be chosen from Linear Multi Variable Regressions, KNNR, Decision Tree, Artificial Neural network, Support Vector Machine, Phantom Traffic modelling, etc. We will also look into details regarding what crowdsourcing, which is our source of data, is.

2.2 Crowdsourcing

Crowdsourcing is a specific sourcing model in which individuals or organizations use contributions from Internet users to obtain needed services or ideas. Crowdsourcing was coined in 2005 as a portmanteau of crowd and outsourcing. This mode of sourcing, which is to divide work between participants to achieve a cumulative result, was already successful prior to the digital age (i.e., "offline"). Crowdsourcing is distinguished from outsourcing in that the work can come from an undefined public (instead of being commissioned from a specific, named group) and in that crowdsourcing, includes a mix of bottom-up and top-down processes. Advantages of using crowdsourcing may include improved costs, speed, quality, flexibility, scalability, or diversity. Crowdsourcing in the form of idea competitions or innovation contests provides a way for organizations to learn beyond what their "base of minds" of employees provides (e.g., LEGO Ideas). Crowdsourcing can also involve rather tedious "microtasks" that are performed in parallel by large, paid crowds (e.g., Amazon Mechanical Turk). Crowdsourcing has also been used for noncommercial work and to develop common goods (e.g., Wikipedia).

Mobile crowdsourcing involves activities that take place on smartphones or mobile platforms, frequently characterized by GPS technology. This allows for real-time data gathering and gives projects greater reach and accessibility. However, mobile crowdsourcing can lead to an urban bias, as well as safety and privacy concerns. In our case, though, the urban bias was what we needed, as the data was Dhaka based.

Crowdsourcing is already used in many places as a popular tool for real time data collection regarding traffic situation.

2.3 Linear Regression

Linear regression is an approach for modeling the relationship between a scalar dependent variable Y and one or more explanatory variables (or independent variables) denoted X . The case of one explanatory variable (independent variable) is called simple linear regression. For more than one explanatory variable (independent variable), the process is called multiple linear regression.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

- If the goal is prediction, or forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed data set of y and X values. After developing such a model, if an additional value of X is then given without its accompanying value of y , the fitted model can be used to make a prediction of the value of y .
- Given a variable y and a number of variables X_1, \dots, X_p that may be related to y , linear regression analysis can be applied to quantify the strength of the relationship between y and the X_j , to assess which X_j may have no relationship with y at all, and to identify which subsets of the X_j contain redundant information about y .

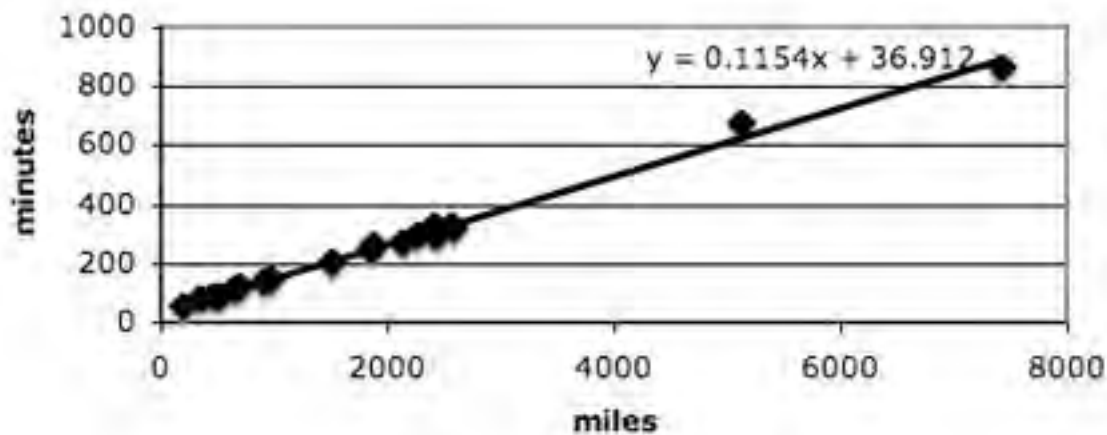


Fig 2.1: Linear Regression

In this representative diagram, time taken is plotted against distance in a scatter graph. The resulting best fit line is created by linear regression and can be used to predict or create a relationship.

A **trend line** represents a trend, the long-term movement in time series data after other components have been accounted for. It tells whether a particular data set (say GDP, oil prices or stock prices) have increased or decreased over the period of time. A trend line could simply be drawn by eye through a set of data points, but more properly their position and slope is calculated using statistical techniques like linear regression. Trend lines typically are straight lines, although some variations use higher degree polynomials depending on the degree of curvature desired in the line.

2.4 K-Nearest Neighbor Algorithm (K-NNR Algorithm)

In pattern recognition, the k-nearest neighbor algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.
- In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the k-NN algorithm is that it is sensitive to the local structure of the data.

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1. Ease to interpret output
2. Calculation time
3. Predictive Power

Let us take a few examples to place KNN in the scale:

	Logistic Regression	CART	Random Forest	KNN
1. Ease to interpret output	2	3	1	3
2. Calculation time	3	2	1	3
3. Predictive Power	2	2	3	2

KNN algorithm fares across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time.

K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$

Manhattan $\sum_{i=1}^k |x_i - y_i|$

Minkowski $\left(\sum_{i=1}^k (|x_i - y_i|^q) \right)^{1/q}$

Fig 2.2: KNN Functions

Algorithm

A simple implementation of KNN regression is to calculate the average of the numerical target of the K nearest neighbors. Another approach uses an inverse distance weighted average of the K nearest neighbors. KNN regression uses the same distance functions as KNN classification.

2.5 Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. It uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

A decision tree consists of three types of nodes:

1. Decision nodes – typically represented by squares
2. Chance nodes – typically represented by circles
3. End nodes – typically represented by triangles

Decision trees are commonly used in operations research and operations management. If, in practice, decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a probability model as a best choice model or online selection model algorithm. Another use of decision trees is as a descriptive means for calculating conditional probabilities.

2.6 Artificial Neural Network

Artificial neural networks (ANNs) or connectionist systems are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using

the analytic results to identify cats in other images. They have found most use in applications difficult to express in a traditional computer algorithm using rule-based programming.

An ANN is based on a collection of connected units called artificial neurons, (analogous to axons in a biological brain). Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream. Further, they may have a threshold such that only if the aggregate signal is below (or above) that level is the downstream signal sent.

Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times.

The original goal of the neural network approach was to solve problems in the same way that a human brain would. Over time, attention focused on matching specific mental abilities, leading to deviations from biology such as backpropagation, or passing information in the reverse direction and adjusting the network to reflect that information.

Neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games, medical diagnosis and in many other domains.

2.7 Support Vector Machines (SVM)

In machine learning, support vector machines (SVMs, also support vector networks are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate

categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

2.8 Phantom Traffic Factor Prediction

A team of MIT mathematicians has developed a model that describes the formation of "phantom jams," in which small disturbances (a driver hitting the brake too hard, or getting too close to another car) in heavy traffic can become amplified into a full-blown, self-sustaining traffic jam. Key to the study is the realization that the mathematics of such jams, which the researchers call "jamitons," are strikingly similar to the equations that describe detonation waves produced by explosions, says Aslan Kasimov, lecturer in MIT's Department of Mathematics. That discovery enabled the team to solve traffic-jam equations that were first theorized in the 1950s.

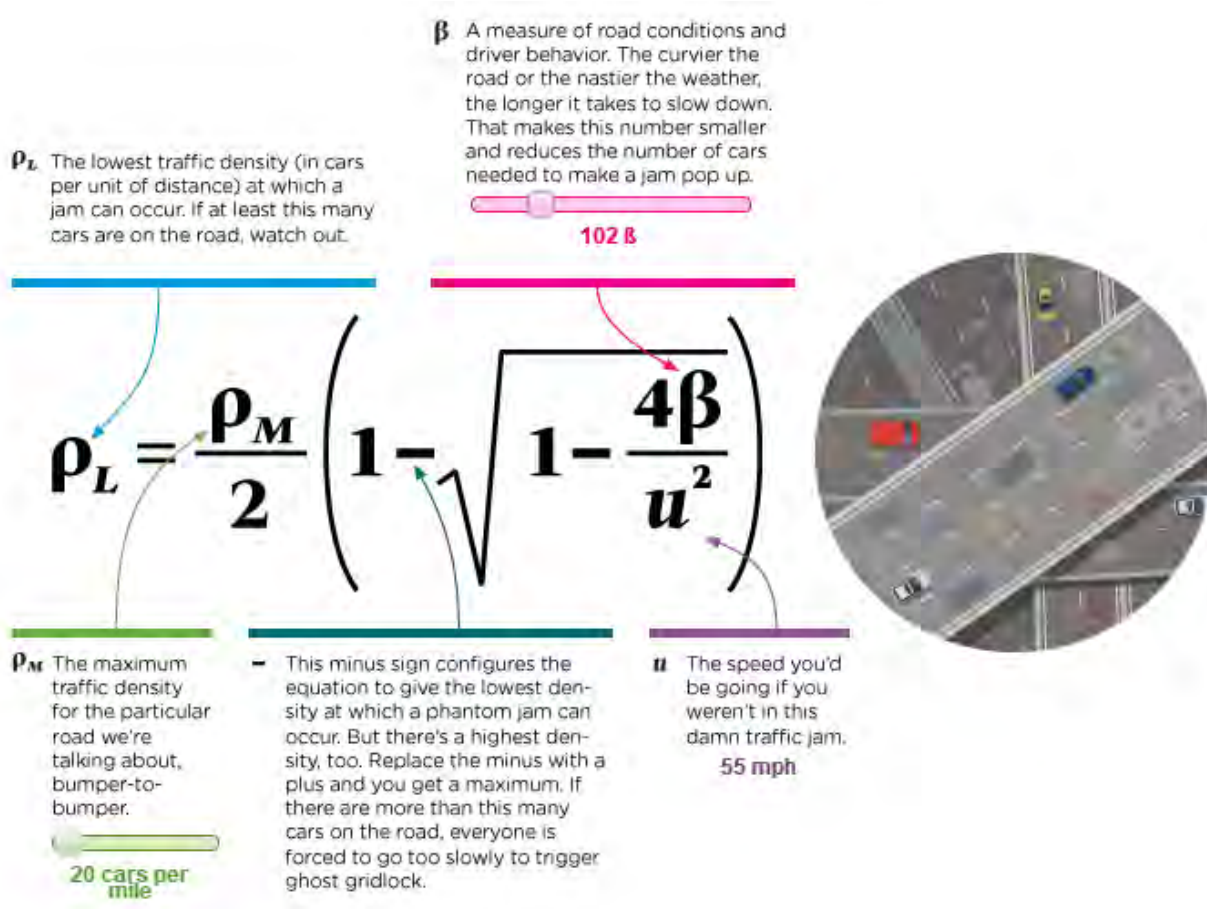


Fig 2.3: Phantom Traffic Factor

Using the equation above could help in predicting and taking into account the phantom factor. However, upon later circumstances, we had to discard Phantom Traffic System.

2.9 FLASK (Webhook/Web Framework)

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It is BSD licensed.

The latest stable version of Flask is 0.12 as of December 2016. Applications that use the Flask framework include Pinterest, LinkedIn, and the community web page for Flask itself.

Flask is called a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

Features

- Contains development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja2 templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired

2.10 Competitor Analysis

DRT is a lite localized app (Dhaka based). Though there are far more sophisticated apps, software or site to provide similar service, it would not be applicable to call them the competitor of DRT. Nevertheless, it is important to analyze the activities of similar ventures in order to get a better

insight in the working procedure or to find out their potential shortcoming and try to improve on it.

2.10.1 Google Maps

Google Maps is a web mapping service developed by Google. It offers satellite imagery, street maps, 360° panoramic views of streets (Street View), real-time traffic conditions (Google Traffic), and route planning for traveling by foot, car, bicycle (in beta), or public transportation.

Like many other Google web applications, Google Maps uses JavaScript extensively. As the user drags the map, the grid squares are downloaded from the server and inserted into the page. When a user searches for a business, the results are downloaded in the background for insertion into the side panel and map; the page is not reloaded. Locations are drawn dynamically by positioning a red pin (composed of several partially transparent PNGs) on top of the map images. A hidden IFrame with form submission is used because it preserves browser history. The site also uses JSON for data transfer rather than XML, for performance reasons. These techniques both fall under the broad Ajax umbrella. The result is termed a slippy map and is implemented elsewhere in projects such as OpenLayers.

In October 2011, Google announced MapsGL, a WebGL version of Maps with better renderings and smoother transitions.

The version of Google Street View for classic Google Maps requires Adobe Flash.

Google Indoor Maps uses JPG, .PNG, .PDF, .BMP, or .GIF, for floor plan.

In 2007, Google Maps began offering traffic data in real-time, using a colored map overlay to display the speed of vehicles on particular roads. Crowdsourcing is used to obtain the GPS-determined locations of a large number of cellphone users, from which live traffic maps are produced.

2.10.2 Go! Traffic

Go! Traffic is an app showing current traffic scenario of Dhaka city. It shows the jam condition by lighting up different intersection with different colored lights to show the traffic intensity, red being the most packed and green being comparatively free road. They have very recently partnered with Robi to provide mobile SMS for traffic updates.

2.10.3 RastaRObosta

RastaRObosta, a real-time traffic launched in 2015, aims to provide user generated information about real time traffic condition in Dhaka so that people can choose less crowded routes. Once a user installs the app it will ask for information about the road that they are on or a road that they are close enough to see. A user can report that information through the app to a central server. Once they input that information, they will see a map of Dhaka city showing all the major roads, including lanes. The users will also see colors for each of these roads in green, yellow and red. Those colors will indicate the condition of the road. Those colors will be determined from the values that other users have selected for that area. The color will be updated frequently so that a user has the most up to date view of road conditions in Dhaka.

Chapter 3.0 System Analysis

After identifying the problem and deciding on a method to approach it by studying existing literature, it is important to identify the working that will be necessary for the project. Its work and information flow must be identified as well as approaching the analytical tools necessary for the raw data to become viable outputs. This chapter deals with the data flows and also the details of algorithms for our chosen machine learning method (ANN).

3.1 Context Diagram

Context diagram shows the interaction between the main portion or functions of the system.

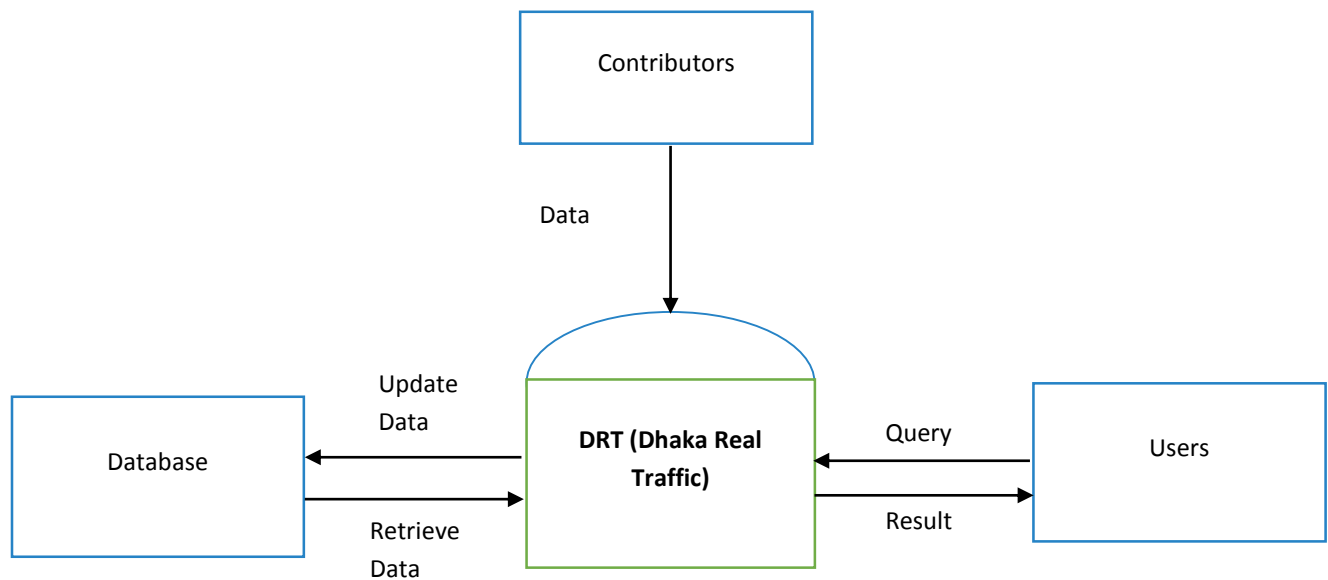


Fig 3.1: Context Diagram

3.2 Decision Tree

The decision tree is a process flow that enables a checking mechanism at each stage. It shows which process comes after which and if they are mandatory. Only mandatory steps are present in the decision tree, as the process aborts whenever the result is 'No' at any node.

3.2.1 Contribution

The decision tree for contribution shows the stages a contributor passes through. Since the user has to provide all the data, it must be a 'Yes' on all the stages for the contribution to be made. However, it is to be noted that these actions do not need to be in this particular order.

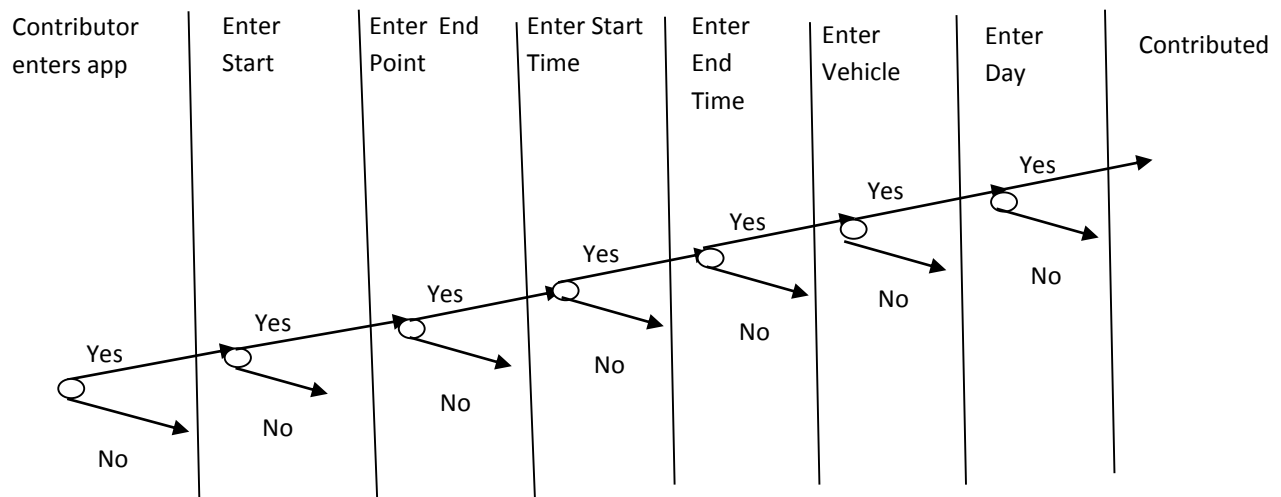


Fig 3.2: Decision Tree for User Contribution

3.2.2 User Query

The decision tree for the user query is similar to that of the contribution, with the same case that the actions in the middle to not need to be in order.

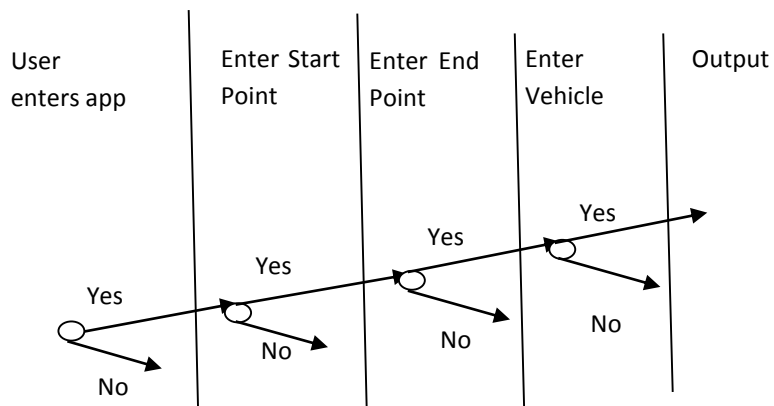


Fig 3.3: Decision Tree for User Query

3.3 Data Flow

The data flow diagram shows the series of activities and if any decision making is involved.

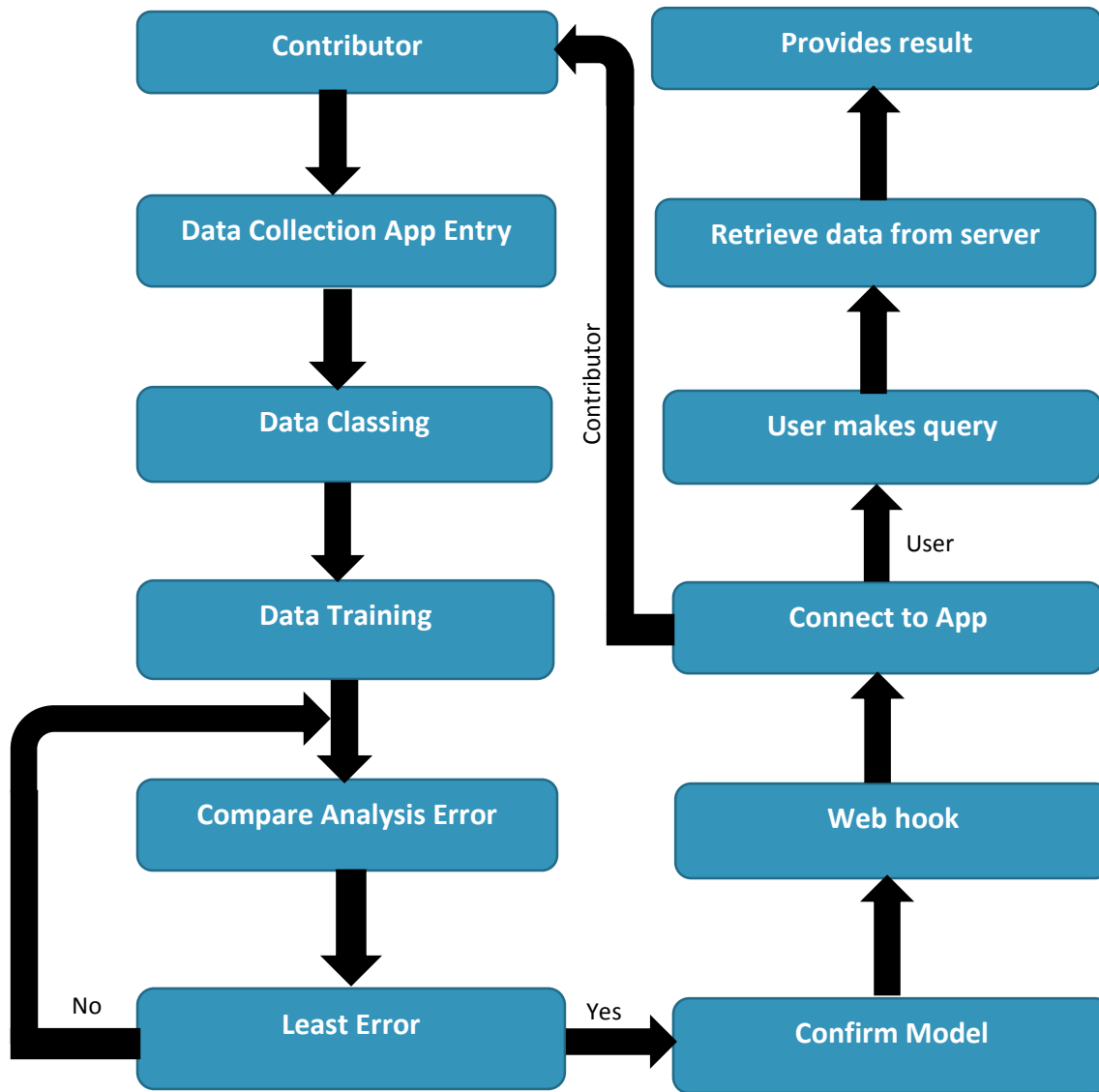


Fig 3.4: Data Flow Diagram

3.4 Error Comparison

The collected data was passed through multiple analytical models. Decision Tree and Linear Multi Variable Regression was rejected previously. SVM, ANN & KNNR modelling was done by Python. The results received are as follows:

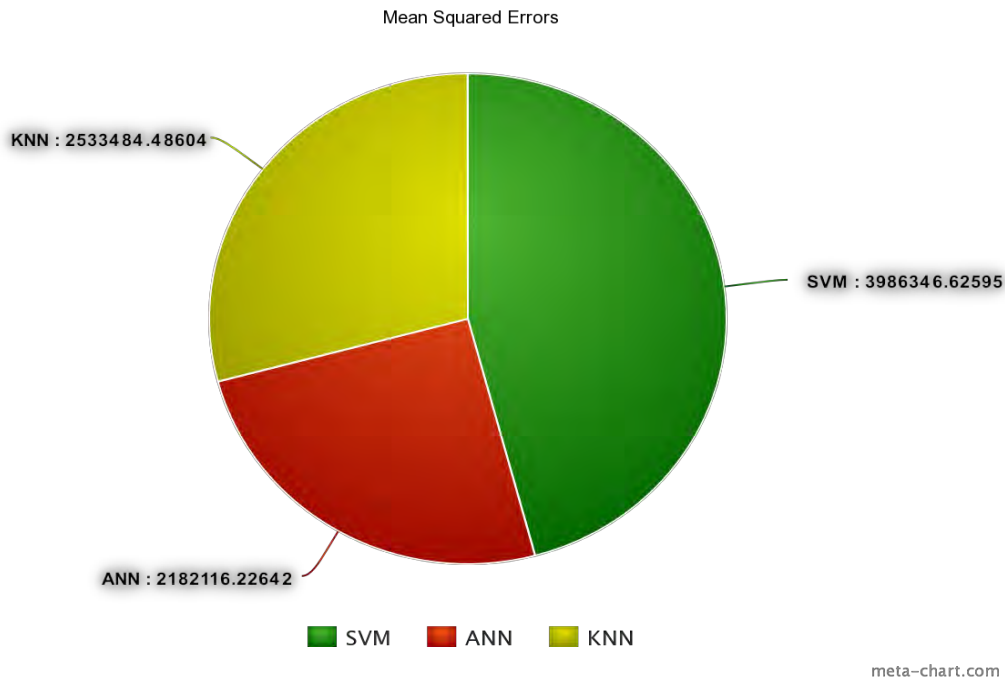


Fig 3.5: Error comparison between machine learning languages

As can be seen from the chart, SVM has the highest error value and is rejected. Though KNN & ANN shows close values, ANN is selected due to the lowest error value of all three of these. Further details regarding the error profile of ANN is shown below in the image.

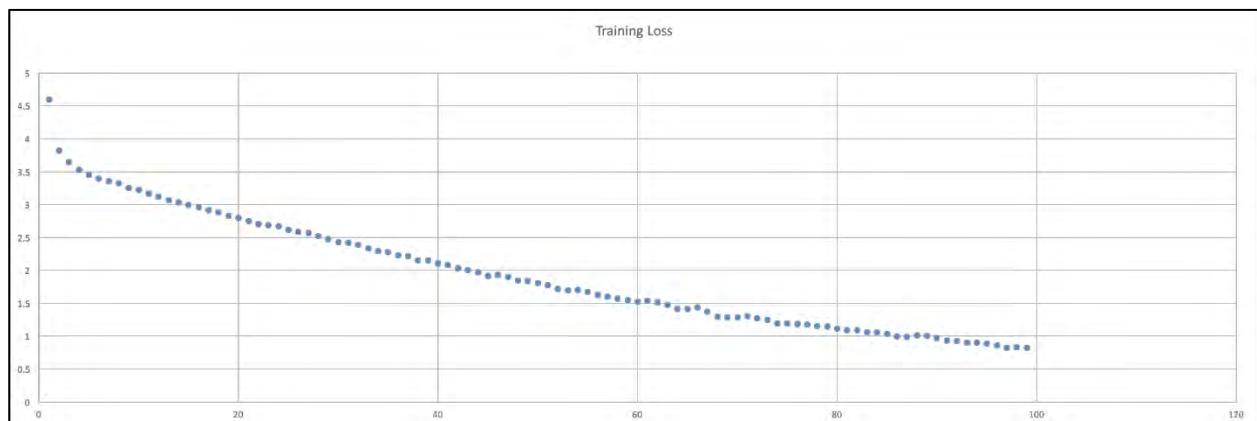


Fig 3.6: ANN error profile showing descent gradient

One benefit drawn from the Neural Network is that its error decreases over each iteration. As can be seen from the graph, the initial error value at low iteration was quite high (4.5+). But as more iterations take place, the error falls over the graph. This is because the ANN teaches itself with more data, thus utilizing machine learning properly.

Moreover, Neural Networking has already been used in multiple similar traffic flow related studies globally. This is considered quite a suitable model for similar types of research. This fact, coupled with the error profiles, has made us chose ANN as our analytical and machine learning model.

3.5 ANN Algorithm

For ANN, gradient descent algorithm will be followed.

Gradient descent, also known as steepest descent, is the simplest training algorithm. It requires information from the gradient vector, and hence it is a first order method.

Let denote $f(w_i) = f_i$ and $\nabla f(w_i) = g_i$. The method begins at a point w_0 and, until a stopping criterion is satisfied, moves from w_i to w_{i+1} in the training direction $d_i = -g_i$. Therefore, the gradient descent method iterates in the following way:

$$w_{i+1} = w_i - g_i \cdot \eta_i, \quad i=0,1,\dots$$

The parameter η is the training rate. This value can either set to a fixed value or found by one-dimensional optimization along the training direction at each step. An optimal value for the training rate obtained by line minimization at each successive step is generally preferable. However, there are still many software tools that only use a fixed value for the training rate.

The next picture is an activity diagram of the training process with gradient descent. As we can see, the parameter vector is improved in two steps: First, the gradient descent training direction is computed. Second, a suitable training rate is found.

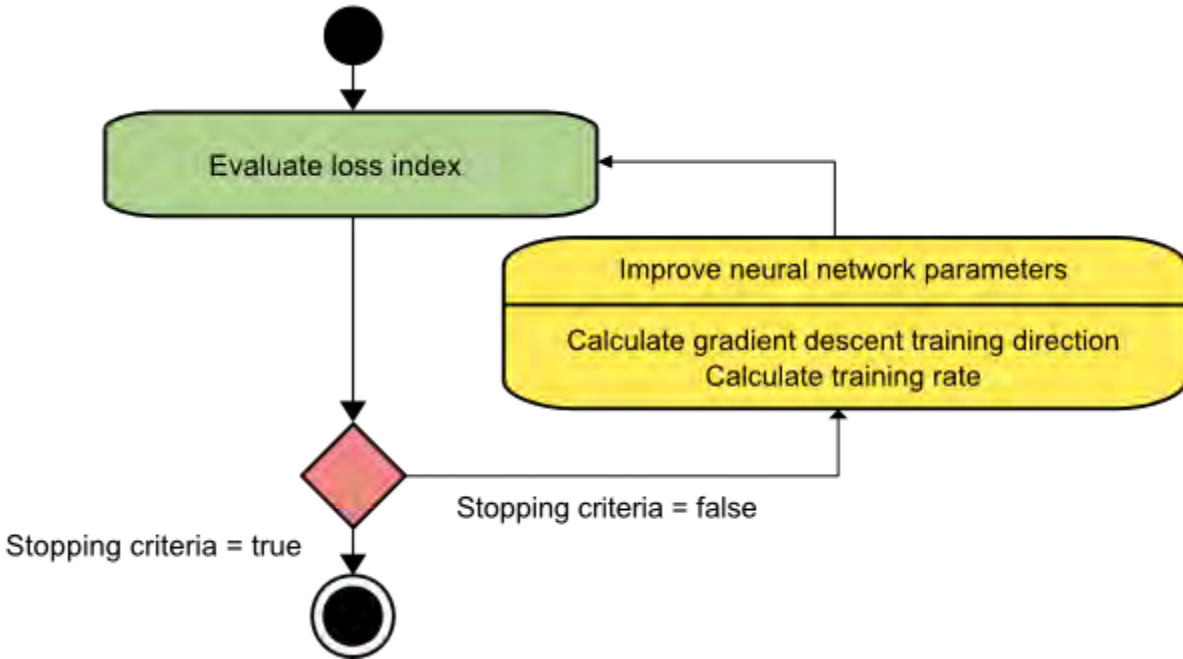


Fig 3.7: Training process with descent gradient

The gradient descent training algorithm has the severe drawback of requiring many iterations for functions which have long, narrow valley structures. Indeed, the downhill gradient is the direction in which the loss function decreases most rapidly, but this does not necessarily produce the fastest convergence. The following picture illustrates this issue.

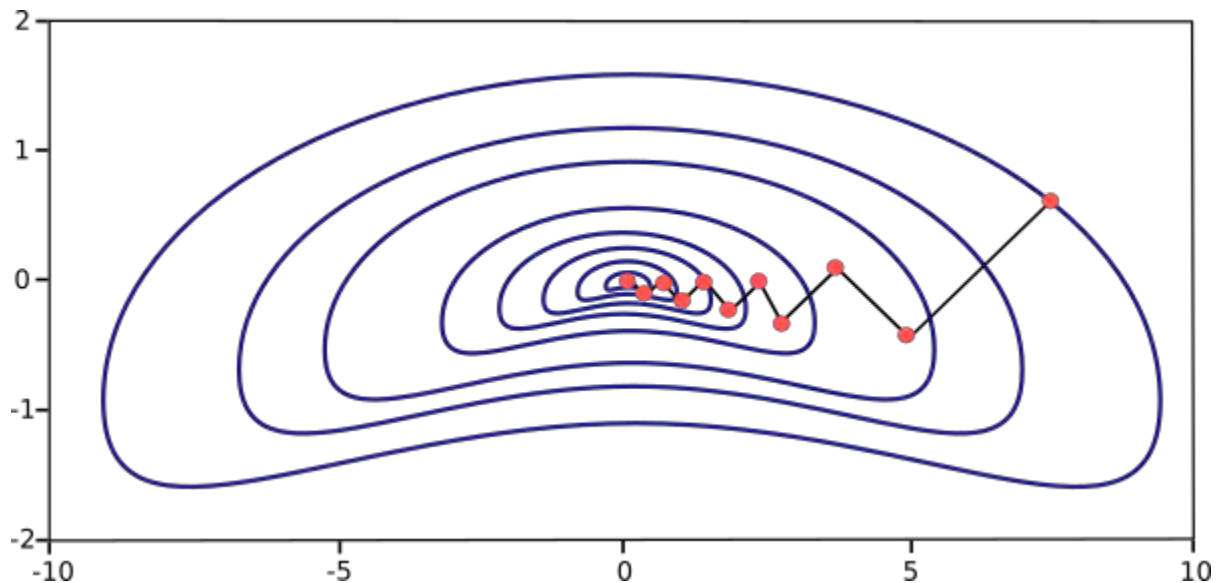


Fig 3.8: Descent Gradient

Gradient descent is the recommended algorithm when we have very big neural networks, with many thousand parameters. The reason is that this method only stores the gradient vector (size n), and it does not store the Hessian matrix (size n^2).

3.6 Result Analysis for Error Comparison

In order to properly identify the best analytical regressor model for DRT, error values were compared between 3 analytical models: KNN, SVM & ANN. In order to properly calculate the error values, we had to give huge effort to streamline our process. This is because, the models will have to deal with multiple variables, such as start point, end point, vehicle, start time, end time, day, duration, etc. If we want to properly display the actual comparison considering all the values in a single diagram, it would become multidimensional. Therefore, we had to fix a certain starting point and consider the duration from that point to each of the other destinations. Therefore, each calculation would define one particular route.

For the sake of representation, Abdullahpur was selected as starting point. The following are the error graphs from Abdullahpur to all other nodes.

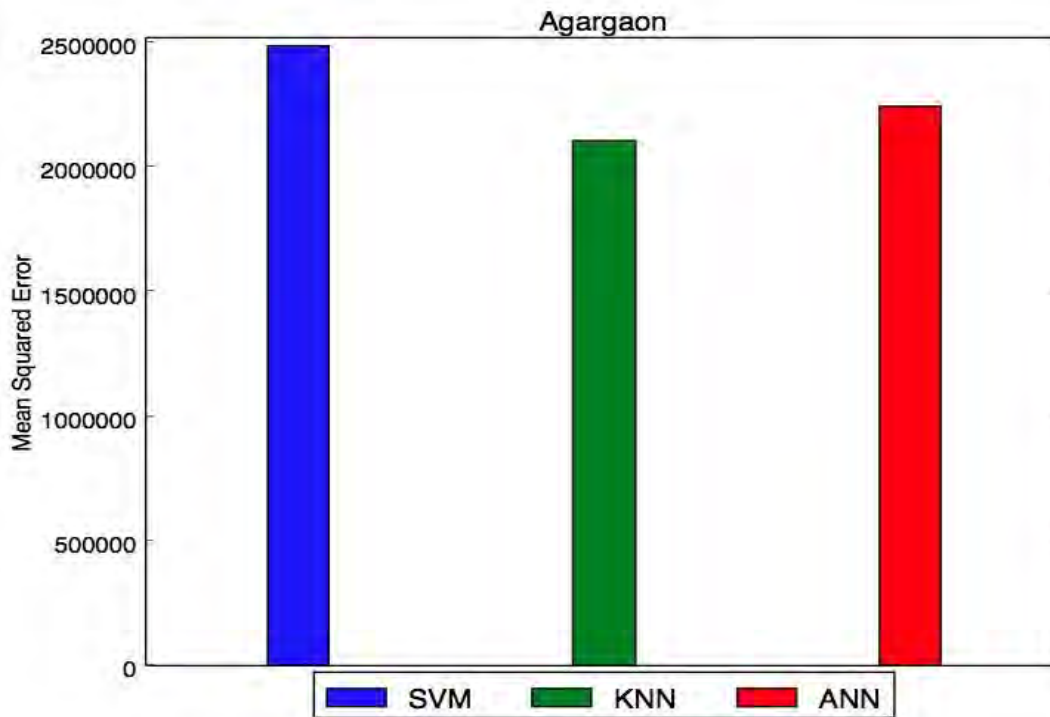


Fig 3.9: Error Comparison between 3 analytical models in Abdullahpur-Agargaon route

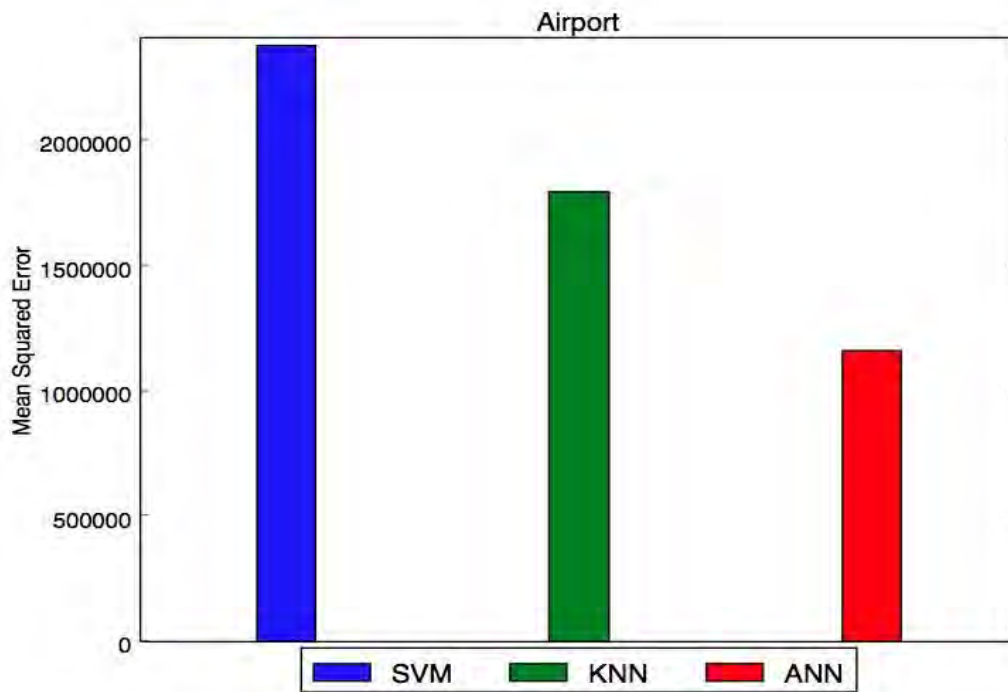


Fig 3.10: Error Comparison between 3 analytical models in Abdullahpur-Airport route

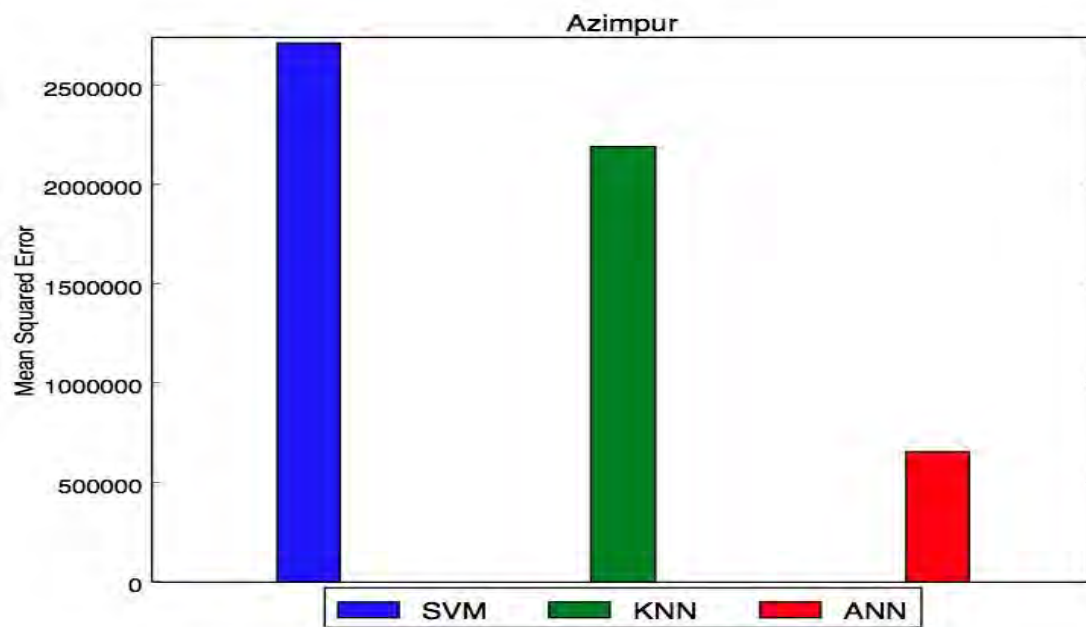


Fig 3.11: Error Comparison between 3 analytical models in Abdullahpur-Azimpur route

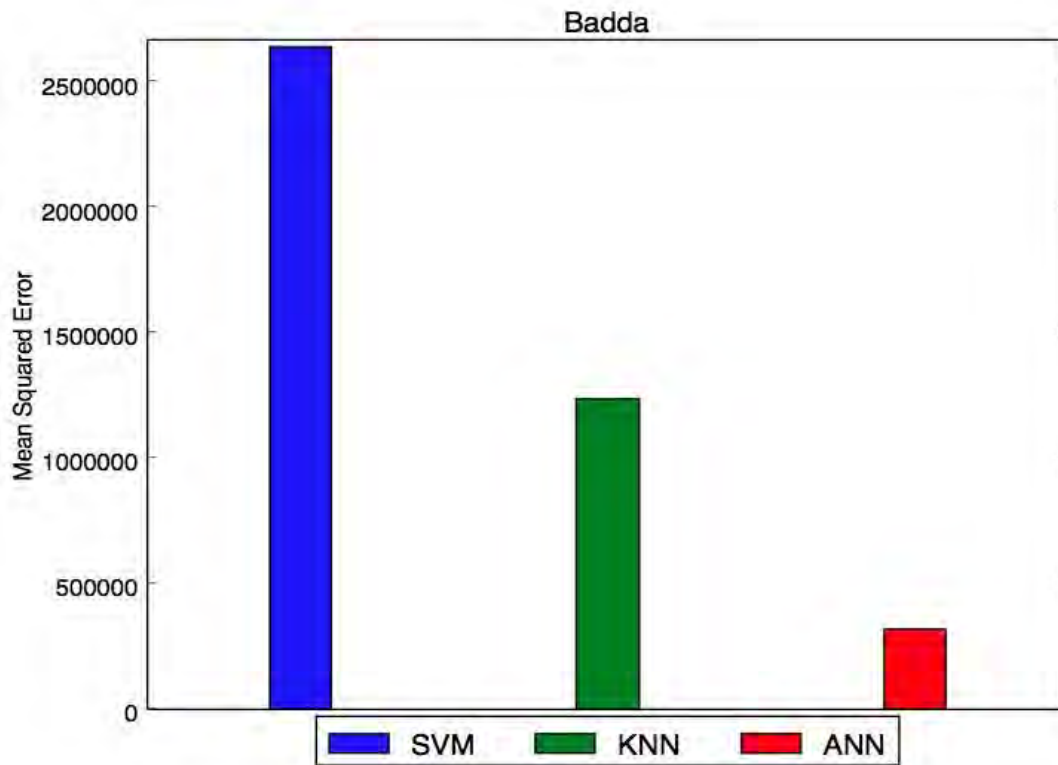


Fig 3.12: Error Comparison between 3 analytical models in Abdullahpur-Badda route

The graphs above show the error found from each of the 3 different methods. From these graphical representations, it is clearly seen that SVM has the higher error among all. With very little exception, ANN provides the least error among all these three. Therefore, it can be surmised that ANN is the best possible method for this analysis.

Chapter 4.0 System Design

After analysis, it is time to decide on the construction of the system. Here, the activities are identified that would give shape to Dhaka Real Traffic. We will also look into the design and architecture of the system by means of pseudo codes and algorithms

4.1 Activity Flow

The system development needs to be planned. During the implementation, the following steps will take place.

- Data Formatting
 - Data classing
 - Data molding
- Prepare train data
 - Input array
 - Output array
- Declare Machine learning (choose from following)
 - K Nearest Neighbor Regression (KNNR)
 - Support Vector Machine (SVM)
 - Neural Network
- Feed training data into machine learning
- Training the model
- Error calculation
- Method selection
- Communication
 - Server prepare FLASK (to communicate with machine language)
 - Define web hook
 - Use Virtual Private Server
- Develop Android App

4.2 Feasibility study

In order to successfully implement the designed app, its viability must be confirmed. Therefore, we conducted some feasibility study in order to confirm its viability.

4.2.1 Technical Viability

DRT will be a lite app. It is technologically very much favorable to the users. The front-end interface is made in a user-friendly way with very simplistic GUI. So, it is very easy to use for everyone. Currently, it is developed only for Android platform. As android smartphones are highly widespread, technological barrier from user point of view is virtually nonexistent. Also, as it is a lite app, there is no high handset configuration requirement. So, every range of smartphone should be able to handle it. Moreover, there will be provision for both online and offline usage. During offline usage, the inbuilt library/database will provide the necessary information. However, internet connectivity will fetch the latest and updated data from the server. Both modes increase the technological flexibility.

The back-end is a different case. As the number of nodes and multi modal relationships are huge in number, we need quite substantial processing power as this app utilized heavy duty programs like Python and Scikit library. A virtual private server will be used in this case to address these issues. Its advantage over a shared server is that there will be flexibility in configuration required and higher RAM & Processor will be available. It will be faster, more efficient and compatible with heavy duty stuff like Python, MC library, etc. These usually do not run in shared server. The downside, however, is the cost.

4.2.2 Financial Feasibility

Currently, the development of the app so far along with future progress is financially feasible. The app is free to use for anybody. However, profit can be generated from advertisements, provided the use of DRT is widespread. As this is crowd-sourced, there was almost no cost involved in the data collection; all were provided voluntarily. The development was done by ourselves, so cost here include our efforts and sweats. As future plans also indicate potential real data entry by users themselves, the financial expense regarding these activities are close to zero. The only costing involved here is the cost of the VPS (Virtual Private Server), which we intend to cover up from advertisement revenue.

4.2.3 Legal Feasibility

There is no legal feasibility issue as there is no breach in any contract, patents or copyrights. Also, it will be free to use.

4.3 Design & Architecture

4.3.1 Pseudo Codes

The pseudo codes are used in multiple levels of DRT development. They provide the initial guidelines or the skeleton on the processes involved and how to proceed.

4.3.1.1 Data Collection

1. Users open app
2. User enter start point from drop down box
3. User enter end point from drop down box
4. User enter start time
5. User enter end time
6. User enter vehicle from drop down box
7. User enter day of the week from drop down box
8. User submits
9. App stores data in backend MS Excel file

4.3.1.2 Backend Data Classing

1. Import all modules and excel files
2. Append all data by field into rows
3. Remove blank data
4. Correct spelling
5. Arrange alphabetically
6. Assign numerical (integer) values to each type of entry in start/end point, vehicle and day (e.g. Abdullahpur is 0, Agargaon is 1 and so on; same for vehicle and day)

4.3.1.3 Data Training

1. Separate all information into 2 separate training array (start point, end point, start time, vehicle and day as input array & end time as output array)
2. Import data into Python to test regressor models
3. Assign different models (ANN, KNN, SVM) to Python
4. Save output and errors
5. Compare errors
6. Choose model with lowest error (ANN for this thesis)

4.3.1.4 Webhook (App Implementation)

1. Import Flask Framework
2. Import Map from Data Processor (Training Model)
3. Initialize application as Flask Instance
4. Define Route as main Webhook
5. Collect time (if not mentioned) and day from android system
6. Collect start point, end point and vehicle from user input
7. Collect start time (optional) from user input
8. Retrieve route time from server
9. Add route time to start time (user input or current time)
10. Return route time and end time via webhook

4.3.2 Algorithm

Algorithms incorporate machine language in the pseudo code. This is a mandatory stage in the development of any app or system.

4.3.2.1 Data Processor

1. IMPORT MODULES
2. IMPORT DATA1.xls, DATA2.xls, DATA3.xls
3. Create ALL_DATA.xls
4. FOR EACH ROW in DATA1.xls, DATA2.xls, DATA3.xls
 APPEND(ALL_DATA.xls, ROW)
5. FOR EACH DATA IN ALL_DATA.xls:
 IF DATA == BLACK_IN_SOME_FIELD:

```

    DEL (DATA)

6. CREATE SORTED.txt FILE

7. FOR EACH ROW IN ALL_DATA.XLS:

    APPEND IF NOT EXISTS ROW['START_PLACE']

8. READ SORTED.TXT

9. SORT(SORTED.TXT)

10. SAVE SORTED.TXT

11. ASSING I=1.0

12. FOR EACH PLACE IN SORTED.TXT:

    MAP PLACE WITH I

    I++

13. INIT WEEKDAYS = ['SATURDAY', 'SUNDAY', 'MONDAY', 'TUESDAY', 'WEDNESDAY',
'THURSDAY', 'FRIDAY']

14. INIT VEHICLES = ['CAR', 'BUS']

15. ASSIGN I = 1

16. FOR EACH WEEKDAY IN WEEKDAYS:

    MAP WEEKDAY WITH I

    I++

17. ASSIGN I = 1

18. FOR EACH VEHICLE IN VEHICLES:

    MAP VEHICLE WITH I

    I++

19. INIT ARRAY TRAIN_X=[], TRAIN_Y=[]

20. FOR EACH ROW IN ALL_DATA.XLS:

    TMP = []

    TMP['START_POS'] = MAP(ROW['START_POS'], SORTED.TXT)

    TMP['STOP_POS'] = MAP(ROW['STOP_POS'], SORTED.TXT)

    TMP['WEEKDAY'] = MAP(ROW['WEEKDAY'], WEEKDAY_MAPPING)

    TMP['VEHICLE'] = MAP(ROW['VEHCILE'], VEHICLE_MAPPING)

    TMP['START_MM'] = SPLIT ROW['START'] AT ':'[0]

```

```

TMP['START_SS'] = SPLIT ROW['START'] AT ':'[1]
TMP['STOP_MM'] = SPLIT ROW['STOP'] AT ':'[0]
TMP['STOP_SS'] = SPLIT ROW['STOP'] AT ':'[1]
DURATION = ROW['STOP']-ROW['START']
APPEND(TRAIN_X, TMP)
APPEND(TRAIN_Y, TMP)
RETURN TRAIN_X, TRAIN_Y

```

4.3.2.2 Data Training

```

1. IMPORT REGRESSOR_MODEL FROM SKLEARN MODULE
2. IMPORT TRAIN_X, TRAIN_Y FROM DATA_PROCESSOR
4. INIT CLF = REGROSSOR MODEL
5. SET HYPERPARAMETERS FOR THE CLF
6. FIT CLF(TRAIN_X, TRAIN_Y)
7. SERIALIZE CLF AND SAVE
8. RETURN CLF

```

4.3.2.3 Webhook

```

1. IMPORT FLASK FRAMEWORK
2. IMPORT MAP FROM DATA PROCERSSOR
2. INIT APP AS FLASK INSTANCE
3. DEINE APP.ROUTE('/') AS MAIN WEBHOOK AS POST METHOD
5. DEFINE TMP = []
4. GET START_POS, STOP_POS, WEEKDAY, TIME FROM POST REQUEST
    TMP['START_POS'] = MAP(START_POS)
    TMP['STOP_POS'] = MAP(STOP_POS)
    TMP['WEEKDAY'] = MAP(WEEKDAY)
    TMP['TIME_MM'] = SPLIT(TIME, ":")[0]
    TMP['TIME_SS'] = SPLIT(TIME, ":")[1]
5. IMPORT CLF FROM TRAINING

```


6. DEFINE SEC = CLF PREDICT(TMP)

7. CALCULATE TIME = CURR_TIME + SEC

8. RETURN TIME THROUGH WEBHOOK

Chapter 5.0 Implementation

After the analysis and design is completed, the implementation or actual app building phase is required. This is twofold in this case. Two apps were developed, one for data collection and another as the final app.

5.1 Data Collection App

The final app is a very data extensive app and requires massive and continuous regular input of actual travel data from commuters. For the initial phase before the DRT was created, a data collection app was created. The app was used for the data collection as this approach is similar to the final app that would be used and commuters would get a real feel and get used in advance. Also, as it can be carried in smartphones, it increases the level of ease for the data entry.

The app created was made by android studio and is supported by all android OS updates. Users enter data like name, starting point, starting time, ending point, ending time, mode of transport, etc. The app transfers these data by JSON parsing to a freehosted website (made by and query support by php platform). The data was planned to be used in various analysis model like regression analysis, phantom traffic prediction, factor analysis, etc. But later, Artificial Neural Network modelling was chosen. Upon completion of analysis, a final app will be created that can forecast the travel time based on the result of analysis as modelled from the data collection app.

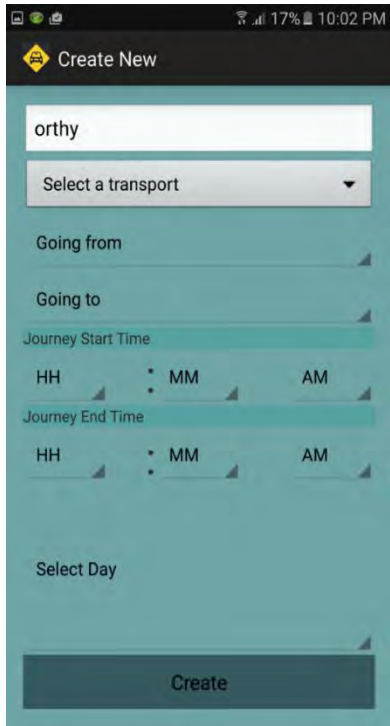


Fig 5.1: Data Collection Entry Screen

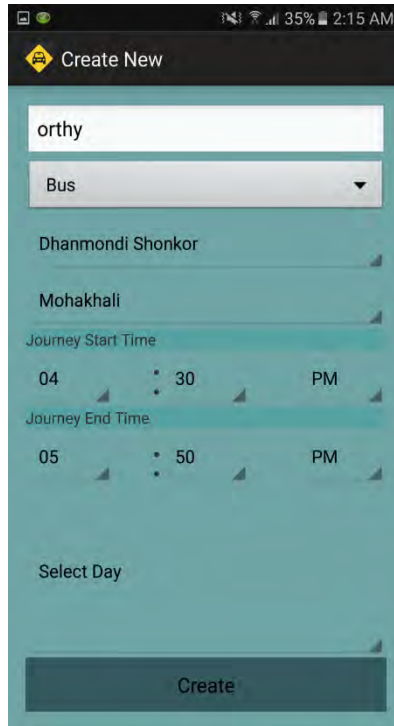


Fig 5.2: Data Collection Completed Screen

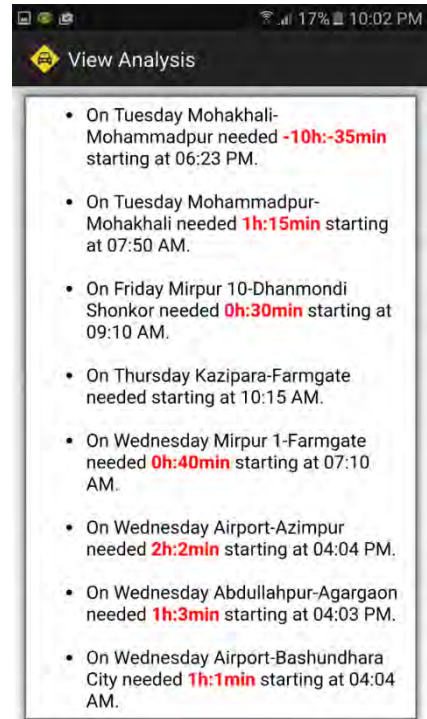


Fig 5.3: Data Collection Auto Generated Summary/Analysis

This creates a database for information on name (later decided redundant), vehicle, start point, end point, start time, end time & day of the week, which will be later used for the analysis.

The collected data is entered JSON Parsing to a free hosted website (Blackjack). This site provides live update of the data whenever provided by a commuter.

Traffic Analysis											Logout
S.N	Name	Transport	Month	Day	From	To	Start time	End time	Total time	Action	
1	nitu	Private Car	July	Sunday	Dhanmondi Shonkor	Kallyanpur	10:03 PM	10:33 PM		Delete	
2	Adhora	Bus	July	Saturday	Farmgate	Agargaon	03:05 PM	03:35 PM	0h:30min	Delete	
3	Adhora	CNG	July	Saturday	Lalbagh	Mohakhali	06:15 PM	07:30 PM	1h:15min	Delete	
4	Adhora	CNG	July	Saturday	Mohakhali	Lalbagh	02:50 PM	03:40 PM	0h:50min	Delete	
5	Adhora	CNG	July	Friday	Jigatola	Mohakhali	04:35 PM	05:01 PM	0h:26min	Delete	
6	Adhora	CNG	July	Friday	Mohakhali	Jigatola	01:20 PM	01:45 PM	0h:25min	Delete	
7	Sadia	CNG	June	Monday	Gulshan 1	Khilgaon	06:01 PM	06:56 PM	0h:55min	Delete	
8	Sunha	Rickshaw	June	Friday	Banani	Farmgate	05:30 PM	06:20 PM	0h:50min	Delete	
9	Shiithi	Private Car	June	Friday	Agargaon	Mirpur 1	05:01 PM	05:33 PM	0h:32min	Delete	
10	Akib	CNG	June	Monday	Malibag	Mirpur 1	03:08 PM	04:25 PM	1h:17min	Delete	
11	Arman	Private Car	June	Friday	Baily Road	Kallyanpur	09:44 PM	10:18 PM	0h:34min	Delete	
12	Arman	Private Car	June	Friday	Kallyanpur	Baily Road	04:02 PM	05:06 PM	1h:4min	Delete	
13	Shafin	Moto Cycle	June	Friday	Badda	Banglamotor	03:04 PM	03:50 PM	0h:46min	Delete	
14	Tipu	Private Car	June	Wednesday	Eskaton Road	Gulshihan	05:04 PM	05:57 PM	0h:53min	Delete	
15	alvi	Moto Cycle	June	Saturday	Azimpur	Eskaton Road	06:08 PM	06:56 PM	0h:48min	Delete	
16	munni	Bus	June	Thursday	Banglamotor	Gulshan 1	08:01 PM	09:35 PM	1h:34min	Delete	
17	Arju	CNG	June	Thursday	Abdullahpur	Dhaka University	02:03 PM	04:35 PM	2h:32min	Delete	
18	irin	CNG	June	Thursday	Agargaon	Jatrabari	05:04 PM	07:04 PM	2h:0min	Delete	
19	tonu	Bus	June	Thursday	Agargaon	Mogbazar	11:10 AM	11:50 AM	0h:40min	Delete	
20	tonu	Bus	June	Monday	Mogbazar	Agargaon	11:01 AM	11:50 AM	0h:49min	Delete	
21	tonu	Bus	June	Sunday	Mirpur 10	Banani	08:10 AM	09:01 AM	0h:51min	Delete	
22	tonu	Bus	June	Saturday	Banani	Mirpur 10	07:10 AM	07:45 AM	0h:35min	Delete	
23	Adhora	Bus	June	Wednesday	Agargaon	Azimpur	06:05 PM	07:20 PM	1h:15min	Delete	
24	ariba	CNG	May	Tuesday	Agargaon	Malibag	05:07 PM	06:49 PM	1h:42min	Delete	
25	ariba	Bus	May	Monday	Gabtolli (Technical)	Rayer Bazar	09:21 AM	10:50 AM	1h:29min	Delete	
26	lima	CNG	May	Monday	Khilgaon	Mirpur 1	08:10 PM	09:56 PM	1h:46min	Delete	

Fig 5.4: Data Collected in free hosted website (Blackjack)

5.2 Mobile Application (DRT)

The mobile app is the final stage as this is where the project is implemented to be used by the commuters (end users).

5.2.1 Backend

5.2.1.1 Backend (Server) specifics

1. Used Python Language
2. Used xlrd library for reading excel
3. Used Pandas for Data Manipulation
4. Use Scikit Learn as machine learning library
5. Used Flask WebFramework
6. Used VPS (Digital Ocean)
7. Used Ubuntu Server 14.04.02 LTS

5.2.1.2 Programming

Initially, the raw data needed to be classed so that training could be done. The coding was done in Sublime Text in Python Language. The data was imported from the excel files and then mapped

(integer assigned to each value). Before mapping, the data needed to be appended. This teaches the program to identify similar data and class them together.

```

49     t2 = timedelta(hours=start_t_h, minutes=start_t_m)
50     duration = (t1-t2).seconds
51
52
53     curr_x.append(start_pos)
54     curr_x.append(stop_pos)
55     curr_x.append(weekday)
56     curr_x.append(vehicle)
57     curr_x.append(start_t_h)
58     curr_x.append(start_t_m)
59     train_X.append(curr_x)
60     train_Y.append(duration)
61 except Exception as e:
62     pass
63
64
65
66
67 #clf = MLPRegressor(activation='logistic', solver='sgd', max_iter=500000, hidden_layer_sizes=(50, 50, 50, 50), verbose=True)
68 clf = KNeighborsRegressor()
69 clf.fit(train_X, train_Y)
70
71
72 def getPrediction(start, stop, time_h, time_m, weekday, vehicle):
73     start_idx = place_mapping[start]
74     stop_idx = place_mapping[stop]
75     weekday_idx = weekday_mapping[weekday]
76     vehicle_idx = vehicle_mapping[vehicle]
77
78     x = []
79     x.append(start_idx)
80     x.append(stop_idx)
81     x.append(weekday_idx)
82     x.append(vehicle_idx)
83     x.append(float(time_h))
84     x.append(float(time_m))
85
86     return str(clf.predict([x])[0])
87
88
89
90

```

Fig 5.5: Appending Data

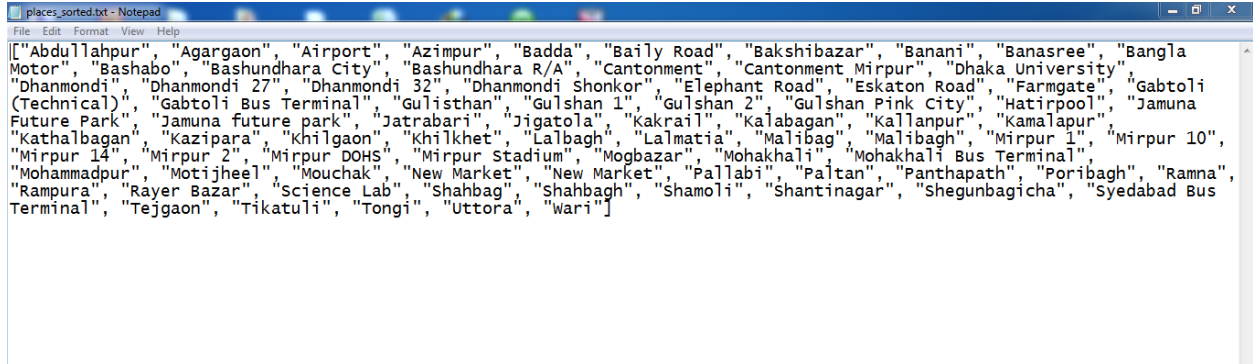
```

File Edit Selection Find View Goto Tools Project Preferences Help
1 #Thesis Project
2
3 from sklearn.neural_network import MLPRegressor
4 from sklearn.neighbors import KNeighborsRegressor
5 import pandas as pd
6 import json
7 from datetime import timedelta
8 from sklearn.metrics import mean_squared_error
9 import math
10
11 df = pd.read_excel('all_data.xlsx')
12 places = json.loads(open('places_sorted.txt').read())
13 ...
14 Sort the all places
15 places = open('places.txt').read().replace('\n', '').split("\n")
16 places.sort()
17 with open('places_sorted.txt', 'w') as f:
18     f.write(json.dumps(places))
19 print places
20 ...
21 place_mapping = {}
22 total_places = len(places)
23
24
25 vehicle_mapping = {'Car' : 0.0, 'Bus' : 1.0}
26 weekday_mapping = {'Saturday' : 0, 'Sunday' : 1, 'Monday' : 2, 'Tuesday' : 3, 'Wednesday' : 4, 'Thursday' : 5, 'Friday' : 6.}
27 for i in range(0, total_places):
28     place_mapping[places[i]] = float(i)
29
30
31 #df['Start_Pos'] = df['Start_Pos'].str.strip().map(place_mapping)
32 #df['Stop_Pos'] = df['Stop_Pos'].str.strip().map(place_mapping)
33 #df['Weekday'] = df['Weekday'].str.strip().map(weekday_mapping)
34 train_X = []
35 train_Y = []
36
37 for index, row in df.iterrows():
38     try:
39         curr_x = []
40         start_pos = place_mapping[str(row['Start_Pos']).strip()]
41         stop_pos = place_mapping[str(row['Stop_Pos']).strip()]
42         weekday = weekday_mapping[str(row['Weekday']).strip()]

```

Fig 5.6: Data mapping for Training

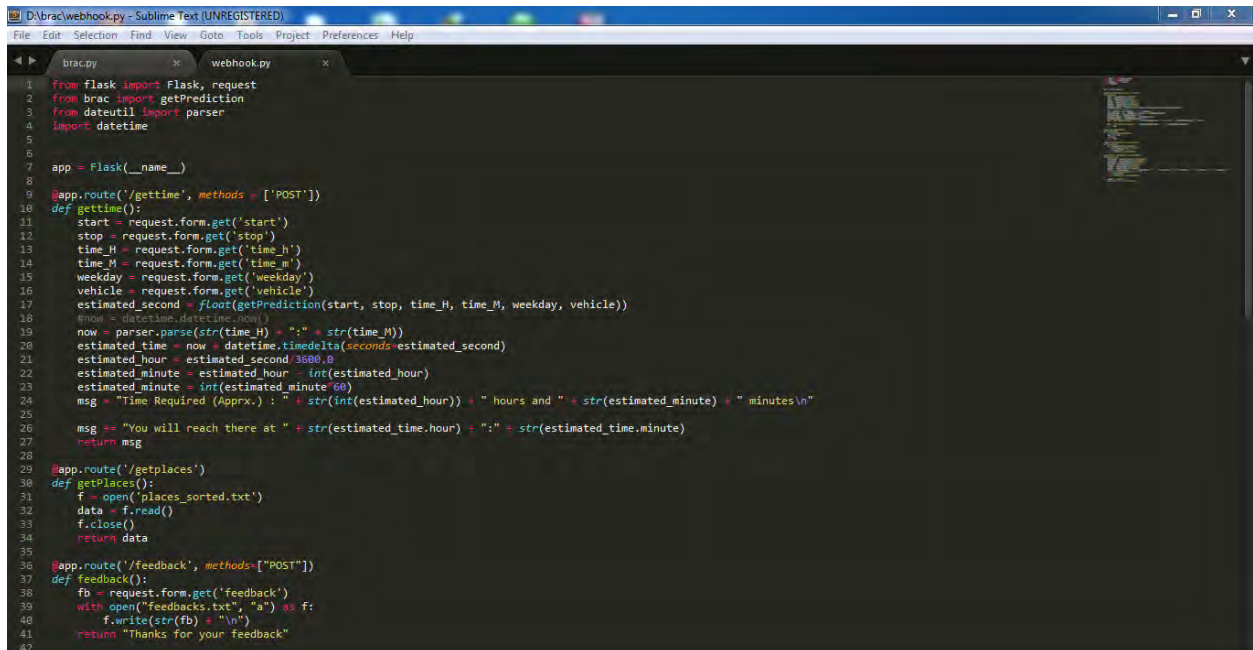
The variety of variables is quite less for vehicles (2 currently) and weekday (7). Their mapping is easy. However, there are currently 78 different destination nodes, which are likely to increase. Therefore, after mapping, they were exported to a notepad file.



```
["Abdullahpur", "Agargaon", "Airport", "Azimpur", "Badda", "Baily Road", "Bakshibazar", "Banani", "Banasree", "Bangla Motor", "Bashabo", "Bashundhara City", "Bashundhara R/A", "Cantonment", "Cantonment Mirpur", "Dhaka University", "Dhanmondi", "Dhanmondi 27", "Dhanmondi 32", "Dhanmondi Shonkor", "Elephant Road", "Eskaton Road", "Farmgate", "Gabtoli (Technical)", "Gabtoli Bus Terminal", "Gulistan", "Gulshan 1", "Gulshan 2", "Gulshan Pink City", "Hatirpool", "Jamuna Future Park", "Jamuna future park", "Jatrabari", "Jigatola", "Kakrail", "Kalabagan", "Kallanpur", "Kamalapur", "Kathalbagan", "Kazipara", "Khilgaon", "Khilkhet", "Lalbagh", "Lalmatia", "Malibag", "Malibagh", "Mirpur 1", "Mirpur 10", "Mirpur 14", "Mirpur 2", "Mirpur DOHS", "Mirpur Stadium", "Mogbazar", "Mohakhali", "Mohakhali Bus Terminal", "Mohammadpur", "Motijheel", "Mouchak", "New Market", "New Market", "Pallabi", "Paltan", "Panthapath", "Poribagh", "Ramna", "Rampura", "Rayer Bazar", "Science Lab", "Shahbag", "Shahbagh", "Shamoli", "Shantinagar", "Shegunbagicha", "Syedabad Bus Terminal", "Tejgaon", "Tikatuli", "Tongi", "Uttora", "Wari"]
```

Fig 5.7: Mapped Location Data

After mapping is completed, the data needs to be trained. All fields other than the 'end time' are considered the 'x' array or the input array. The 'end-time' field is considered the 'y' or output array. The training considers all available parameters of x to provide y. In the below screenshot, our trial with the KNN method is shown. Similarly all three methods were executed to find out the one with least error.



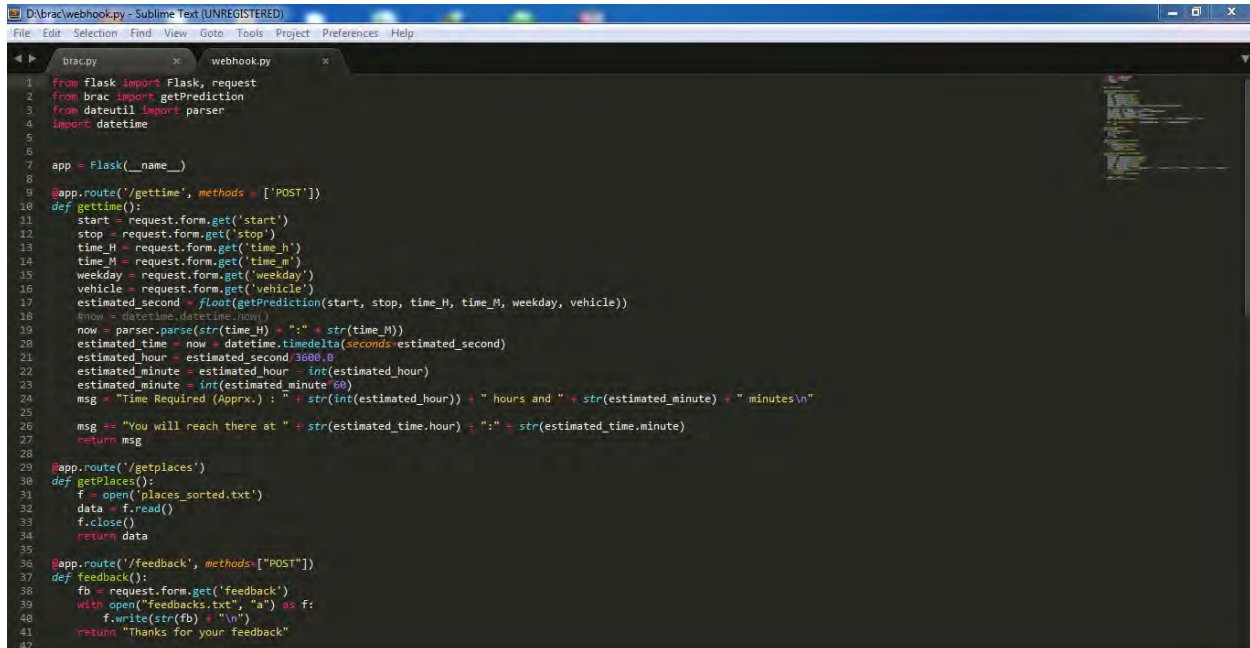
```

1 from flask import Flask, request
2 from brac import getPrediction
3 from dateutil import parser
4 import datetime
5
6
7 app = Flask(__name__)
8
9 @app.route('/gettime', methods = ['POST'])
10 def gettime():
11     start = request.form.get('start')
12     stop = request.form.get('stop')
13     time_H = request.form.get('time_h')
14     time_M = request.form.get('time_m')
15     weekday = request.form.get('weekday')
16     vehicle = request.form.get('vehicle')
17     estimated_second = float(getPrediction(start, stop, time_H, time_M, weekday, vehicle))
18     now = datetime.datetime.now()
19     now = parser.parse(str(time_H) + ":" + str(time_M))
20     estimated_time = now + datetime.timedelta(seconds=estimated_second)
21     estimated_hour = estimated_second/3600.0
22     estimated_minute = estimated_hour - int(estimated_hour)
23     estimated_minute = int(estimated_minute*60)
24     msg = "Time Required (Approx.) : " + str(int(estimated_hour)) + " hours and " + str(estimated_minute) + " minutes\n"
25
26     msg += "You will reach there at " + str(estimated_time.hour) + ":" + str(estimated_time.minute)
27     return msg
28
29 @app.route('/getplaces')
30 def getPlaces():
31     f = open("places_sorted.txt")
32     data = f.read()
33     f.close()
34     return data
35
36 @app.route('/feedback', methods=["POST"])
37 def feedback():
38     fb = request.form.get('feedback')
39     with open("feedbacks.txt", "a") as f:
40         f.write(str(fb) + "\n")
41     return "Thanks for your feedback"
42

```

Fig 5.8: Data Training

When data training is completed, the webhook is prepared. This enables the frontend to communicate with the backend to retrieve data and also to provide input (contribution) when required.



```

D:\brac\wehook.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

brac.py  wehook.py
1  from flask import Flask, request
2  from brac import getPrediction
3  from dateutil import parser
4  import datetime
5
6
7  app = Flask(__name__)
8
9  @app.route('/gettime', methods = ['POST'])
10 def gettime():
11     start = request.form.get('start')
12     stop = request.form.get('stop')
13     time_H = request.form.get('time_h')
14     time_M = request.form.get('time_m')
15     weekday = request.form.get('weekday')
16     vehicle = request.form.get('vehicle')
17     estimated_second = float(getPrediction(start, stop, time_H, time_M, weekday, vehicle))
18     #now = datetime.datetime.now()
19     now = parser.parse(str(time_H) + ":" + str(time_M))
20     estimated_time = now + datetime.timedelta(seconds=estimated_second)
21     estimated_hour = estimated_second // 3600
22     estimated_minute = estimated_hour * 60
23     estimated_minute = int(estimated_minute)
24     msg = "Time Required (Apprx.) : " + str(int(estimated_hour)) + " hours and " + str(estimated_minute) + " minutes\n"
25
26     msg += "You will reach there at " + str(estimated_time.hour) + ":" + str(estimated_time.minute)
27     return msg
28
29 @app.route('/getplaces')
30 def getPlaces():
31     f = open('places_sorted.txt')
32     data = f.read()
33     f.close()
34     return data
35
36 @app.route('/feedback', methods=["POST"])
37 def feedback():
38     fb = request.form.get('feedback')
39     with open("feedbacks.txt", "a") as f:
40         f.write(str(fb) + "\n")
41     return "Thanks for your feedback"
42

```

Fig 5.9: Webhook Coding

5.2.2 Frontend (Mobile App)

5.2.2.1 Frontend specifics

1. Support Upto Android 7.0
2. Used Java as Programming Language
3. Used XML for designing
4. App Communicates with the server using Android Async HTTP Client by loopj library
5. Used Build tool 26

5.2.3 The App in Action

The App is supported in Android based smartphones only. It will soon be available for download from the Google Play store. The App has a small size of 8MB and uses very insignificant amount of memory as all works are done at the backend.

The GUI has been kept very simple to keep the app lite and also in order to deliver the exact information required by the users without unnecessary cluttering.

5.2.3.1 Interface

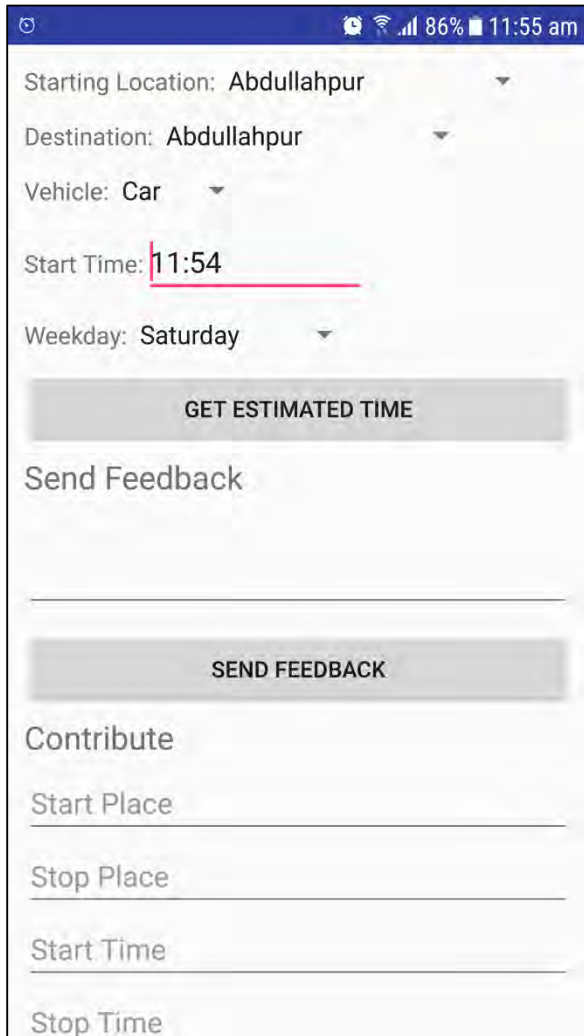


Fig 5.10: Open Screen

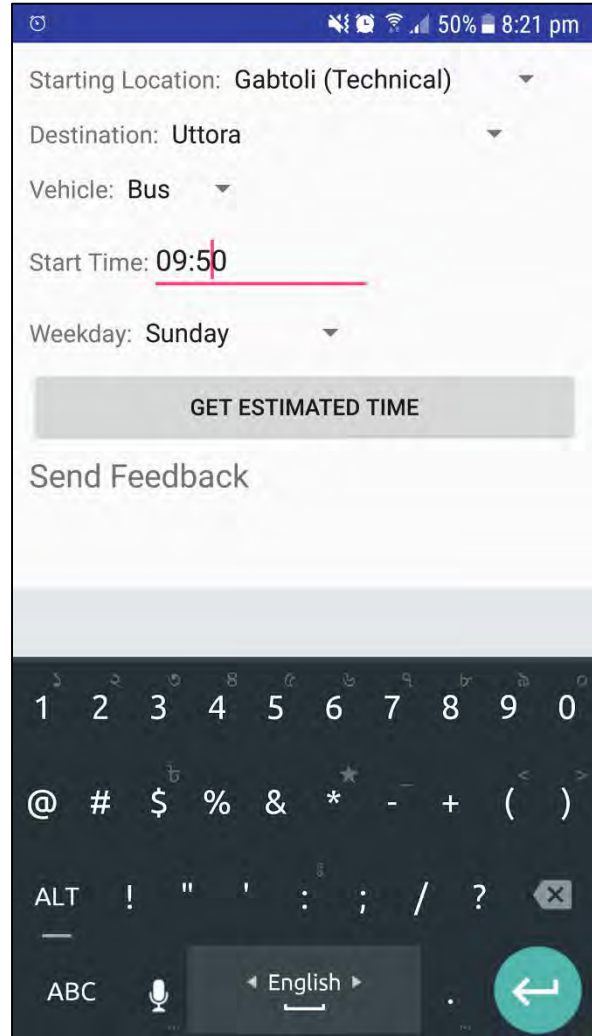


Fig 5.11: Completed Query

The open screen (Fig 5.10) is very simplistic and it shows the required information for the result (Starting Location, Destination, Vehicle, Start Time & Day of Travel). The default time is the current time, collected from the system. The default locations for both the start and end are provided as Abdullahpur, the first in the alphabetic location list. Once all the information is provided, the output will be provided. The next figure (Fig 5.11) shows a complete set of input data required for the output. The Start location, destination, vehicle and weekday can be selected from a drop down box. If the user do not wish to travel at the moment of using the app,

5.2.3.2 Outputs

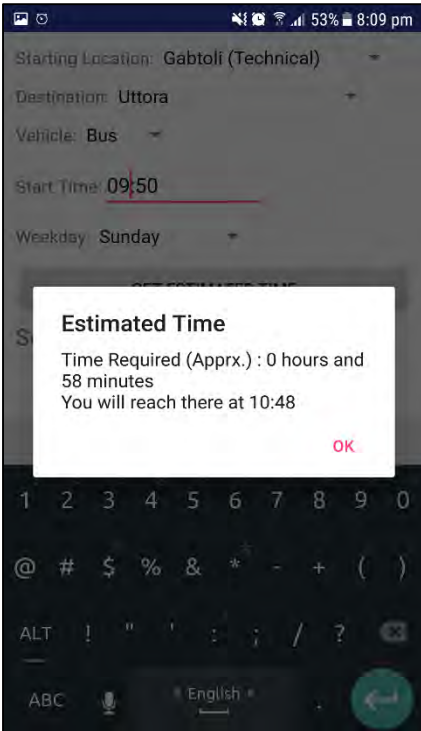


Fig 5.12: Output 1

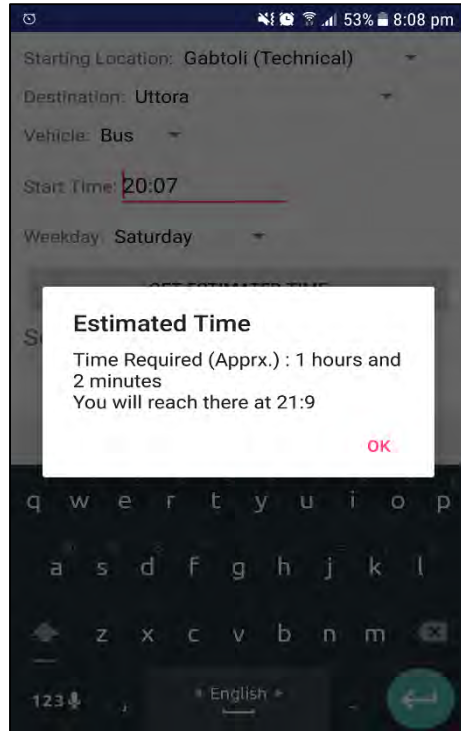


Fig 5.13: Output 2

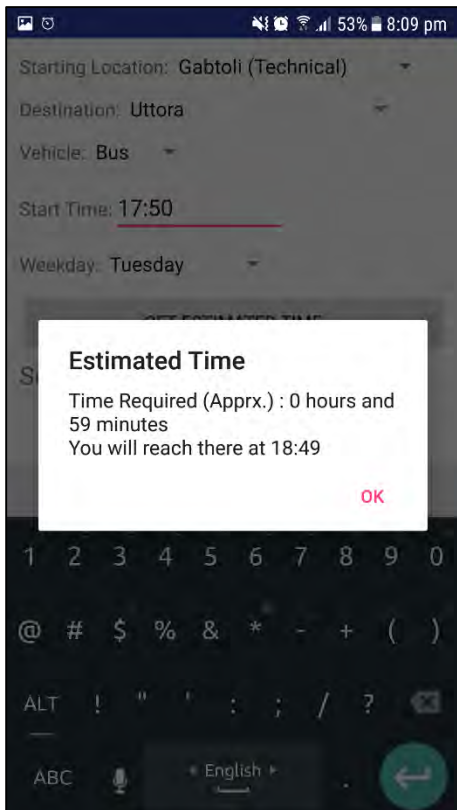


Fig 5.14: Output 3

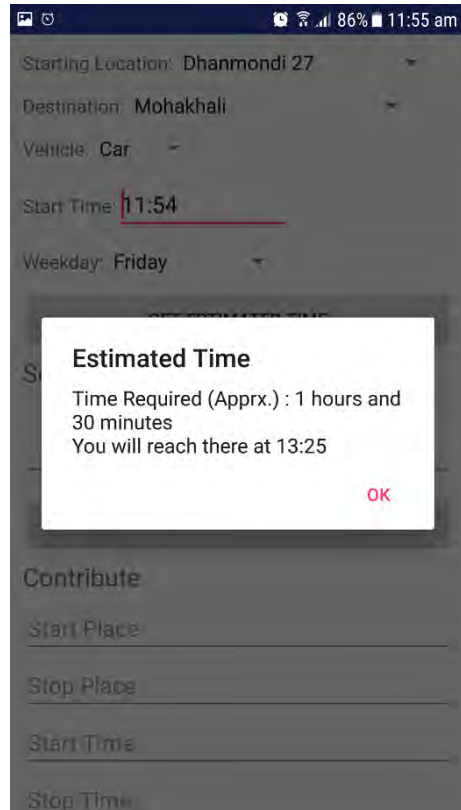


Fig 5.15: Output 4

As can be seen from the figures (Fig 5.12-5.15), the output of the query is given by the duration of the journey and also the end time with consideration with the start time. Also, as can be seen from the screenshots (Fig 5.12-5.14), the estimated time varies for different start time and day for the same route. Fig 5.15 shows the time estimate for a different route.

5.2.3.4 Error validation

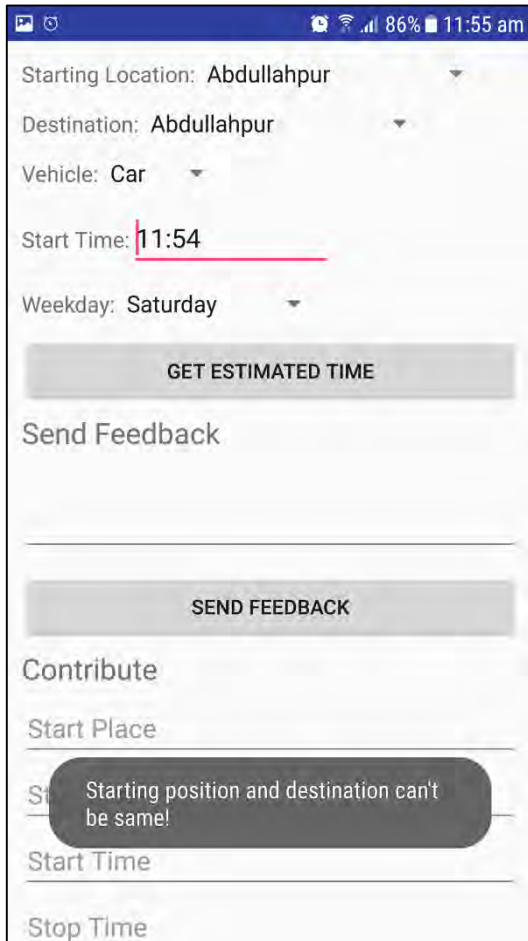


Fig 5.16: Location Error Validation

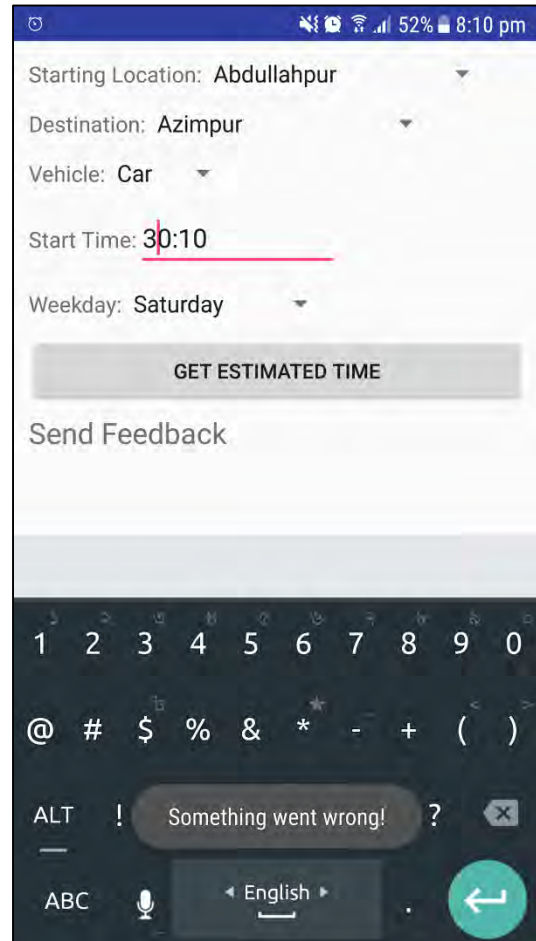


Fig 5.17: Time Error Validation

There are 2 error validations in use. Fig 5.16 shows the location error validation. If the user enters the same start and destination location, the app shows an error message (“Starting position and destination can’t be same!”) and do not provide any output. Fig 5.17 shows time validation. If the user inputs abnormal time (such as hour more than 23 or minute more than 60), then an error message is shown (Something went wrong!) and no output is provided.

5.2.3.5 Feedback

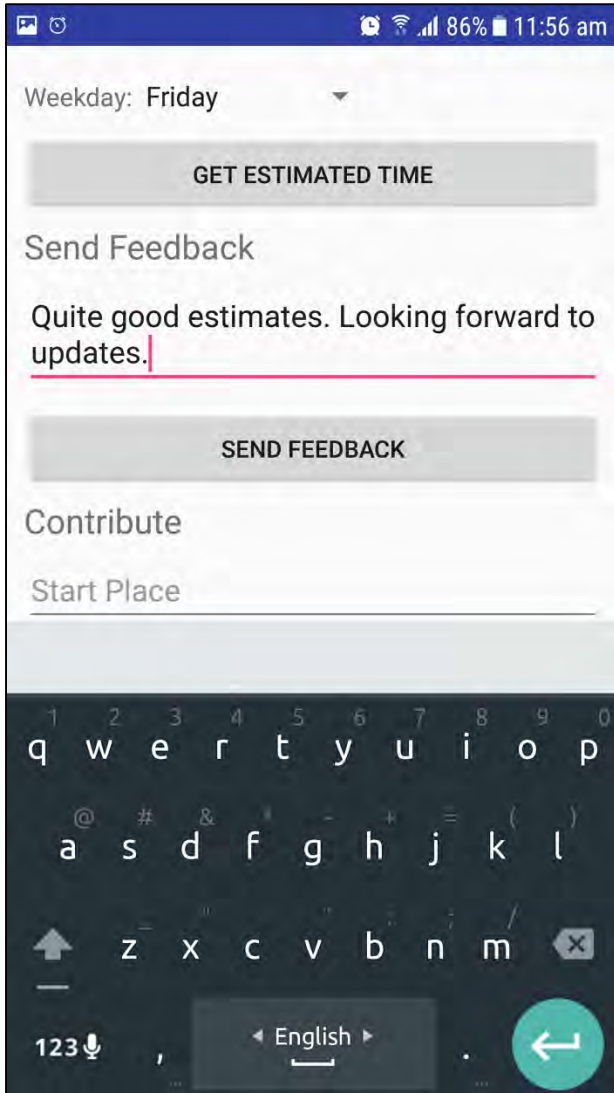


Fig 5.18: Feedback Entry

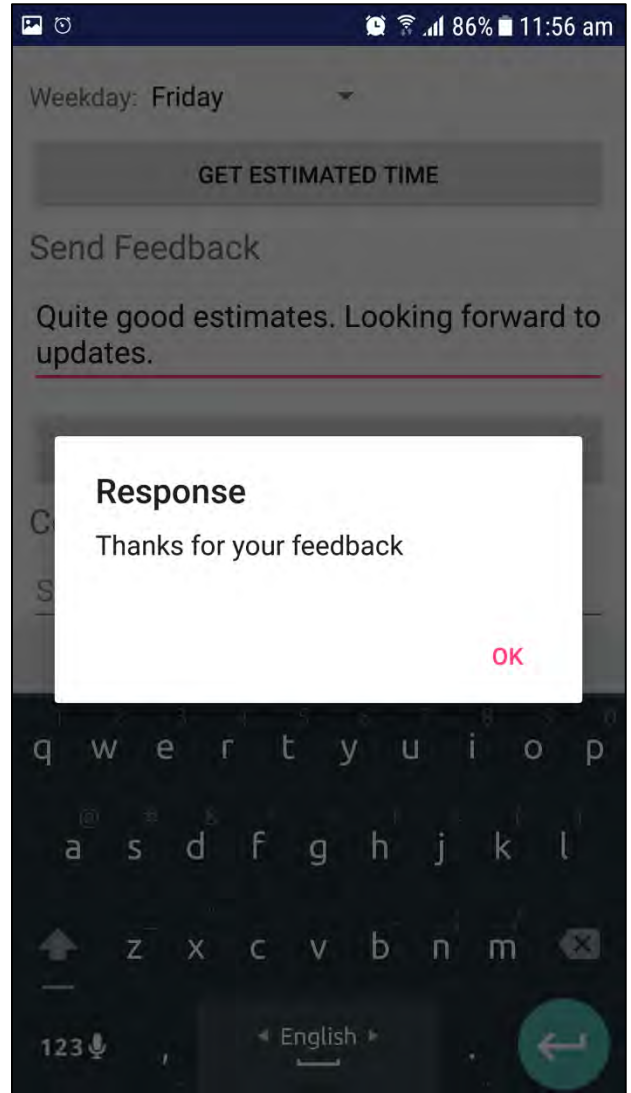


Fig 5.19: Feedback Acceptance

The figures show the Feedback option of the app. Fig 5.18 shows the Feedback entry sample and Fig 5.19 shows the response from the app after accepting the feedback.

5.2.3.6 Contribution

Weekday: Saturday

GET ESTIMATED TIME

Send Feedback

SEND FEEDBACK

Contribute

Mohakhali

Jigatola

10:40

11:50

Car

Sunady

CONTRIBUTE

Fig 5.20: Contribution Entry

Weekday: Saturday

GET ESTIMATED TIME

Send Feedback

SEND FEEDBACK

Response

Thanks for you contribution!

OK

CONTRIBUTE

Fig 5.21: Contribution Acceptance

The figures show the contribution actions. Fig 5.20 shows the filled up contribution fields. Fig 5.21 shows the response message from the app after contribution. In case of contribution, the data entered updates the database at the back end. Drop down boxes are not provided to accommodate for entries of new locations. During the construction of the app, 78 locations were taken inside Dhaka. However, more locations may be frequented. Therefore, if a new location or erroneous data is provided, it can be entered into the database and also the query drop down box upon decision by the admin. If the location is similar to existing one, it will be used to enter the data in that record. Otherwise a new one will be created.

Chapter 6.0 Conclusion

Even though the app has been completed, it is still in an infancy stage. We conclude by describing our aspirations and what we have in mind to develop in the near future.

6.1 Conclusion

Dhaka city continues to remain as one of the most densely populated with the least amount of urban planning going into its rapid growth. As a result, traffic jam continues to stay as one of the most detrimental situation to its citizens. As the situation gets more critical, often the question of how much time is required to travel to the destination is outclassed by the question if one should even travel at all. Our app would, at its current level, be very suitable to provide logical grounds to choose one's schedule considering the traffic scenario.

Many products concerning traffic prediction already exists. However, this paper presents an initial version of a system that ultimately becomes the best source of traffic information providing authentic and realistic data for Dhaka city. Huge potential remains for this project and we expect to take into consideration all potential development to establish it as the best traffic prediction solution for the case of Dhaka City.

The app is still in its infancy stage. Being dependent on crowdsourcing, it needs regular input of huge data. However, as more data is incorporated, its level of accuracy rockets. There are a lot of future development plans in mind. With appropriate implementation, this could become the traffic portal that our citizen needs the most.

6.2 Future planned developments

- **Map integration & GPS:** In future, we aim to let users choose their start and destinations from a Dhaka City map. The machine learning will be taught to take the nearest node from the map, incorporate into database for learning and provide suitable regression. Also, data collection can be done by GPS.
- **Route selection:** Currently, the outcome/duration of the journey is from the average duration of users' journey. However, users may take any of the available routes. E.g. a commuter can go from Abdullahpur to Agargaon via Banani or via Kalshi. The distance and traffic scenario is different for each case. In future, detailed information will be available with users being able to select routes of their commute.
- **Route suggestion:** Based on updated and more detailed crowdsourcing, DRT will be able to suggest the users on which route to follow for better timings when the start & end points are selected.

Bibliography

- [1] T. Hasan, "Development of Transport Models to Predict Traffic Flo-Pattern and Volume of Traffic on the National Highway Network," BUET, Dhaka, 1993.
- [2] K. & Vanajakshi, "Short-term traffic flow prediction using seasonal ARIMA model with limited input data," Chennai, 2015.
- [3] D. Reporter, "Traffic jam in Dhaka eats up 3.2m working hours a day," The Daily Sun, Dhaka, 2017.
- [4] M. Nielsen, "Using neural nets to recognize handwritten digits," in *Neural Networks and Deep Learning*, Determination Press, 2015.
- [5] M. P. M. Raeesi, "Traffic Time Series Forecasting by Feedforward Neural Network: A Case Study Based on Traffic Data of Monroe," ISPRS, Tehran, 2014.
- [6] "Decision Tree," [Online]. Available: https://en.wikipedia.org/wiki/Decision_tree. [Accessed 04 July 2017].
- [7] "Crowdsourcing," [Online]. Available: <https://en.wikipedia.org/wiki/Crowdsourcing>. [Accessed 21 June 2017].
- [8] "Artificial Neural Network," [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network. [Accessed 25 June 2017].
- [9] "Equation: Factors for Predicting Phantom Traffic Jams," [Online]. Available: https://www.wired.com/2010/06/st_equation_traffic/. [Accessed 02 July 2017].
- [10] "Flask (Web Framework)," [Online]. Available: [https://en.wikipedia.org/wiki/Flask_\(web_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework)). [Accessed 12 July 2017].
- [11] "Google Maps," [Online]. Available: https://en.wikipedia.org/wiki/Google_Maps. [Accessed 28 May 2017].
- [12] "KNN Regression," [Online]. Available: http://www.saedsayad.com/k_nearest_neighbors_reg.htm. [Accessed 15 June 2017].
- [13] "Linear Regression," [Online]. Available: https://en.wikipedia.org/wiki/Linear_regression. [Accessed 31 May 2017].
- [14] "Support Vector Machine," [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine. [Accessed 03 July 2017].