



Predictive Location Generation Using Machine Learning Approach

By

S.A.M KHALED HASAN

14201012

SHAHRIAR AL NAHIAN

13301036

SAKIBUL ISLAM

13101181

Department of Computer Science & Engineering

BRAC UNIVERSITY

Supervised by: Dr. AMITABHA CHAKRABARTY

Thesis 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

DECLARATION

We, hereby declare that this thesis is based on results we have found ourselves. Materials of work from researchers conducted by others are mentioned in references.

Signature of Supervisor

Signature of Authors

Dr. Amitabha Chakrabarty
Assistant Professor
Department of Computer Science and
Engineering
BRAC University

S.A.M KHALED HASAN
(14201012)

SHAHRIAR AL NAHIAN
(13301036)

SAKIBUL ISLAM
(13101181)

ABSTRACT

Predicting future locations by analyzing past data is not new. Giants like Google, Yahoo or Amazon all use their own machine learning algorithms to generate it but these companies use the data for business purposes as a result our personal and valuable data get sacrificed for their sake. In our project, Predictive Location Generation Using Machine learning, we have developed a mobile application which can automatically locate present position of users using GPS and then based on these collected data it can generate meaningful predictive future locations using machine learning algorithm. The application then trades these generated locations to extract information (such as traffic updates, crime report of certain regions, addresses of important places, product offers and so on) from third party servers using Internet but the key part of this procedure would be not to disclose the user's identity to the third party.

ACKNOWLEDGEMENT

All the thanks and gratitude goes to Almighty ALLAHU SUBHANU WA TAWALA who is the creator of this vast universe and the source of all kind of knowledge and intelligence, the most merciful, gracious, who give us the strength, guidance, patience and abilities to complete the thesis.

We are especially grateful and thankful to Amitabha Chakrabarty, Ph.D, our thesis supervisor, for his guidance, support, inspiration in completion of our thesis work. Without his proper guidance and support, we could not complete our thesis work.

Besides this, we are thankful to Prema Dev, Samrin Sultana Tithi and Sabiha Islam Aunamika, the author of “Crime Mapping through Digital Data Analysis from Intermediate Repository by Crowd Sourcing”, for their support to resolve problems about data retrieving from external server related issues.

We are also grateful to Faculties, Staffs of Computer Science & Engineering Department, BRAC University for enlightening us with the proper knowledge and helping us throughout our academic period in BRAC University.

Finally, we are expressing our sincere gratitude to our beloved parents who have supported us financially, tactically and mentally. We would also like to express our gratefulness to our brothers, sisters and our friends for their supports, love and encouragement.

Table of Contents

List of Figures.....	VI
List of Tables.....	VIII
CHAPTER 1: INTRODUCTION.....	1-2
1.1 Motivation.....	1
1.2 Objectives & Goals.....	1
1.3 Thesis Orientation.....	2
CHAPTER 2: LITERATURE REVIEW.....	3-5
2.1 PLG	3
2.2 Logistic Regression.....	4
2.3 Decision Tree.....	4
2.4 k-NN.....	4
2.5 Related Works and Research.....	5
CHAPTER 3: METHODOLOGY.....	6-17
3.1 Predictive Location Generation Using Machine Learning Approach	6
3.2 Logistic Regression.....	7
3.2.1 How to Implement	9
3.3 Decision Tree.....	12
3.3.1 Pseudo-code Implementation of Decision Tree.....	12
3.4 k-NN.....	13
3.4.1 Calculate Distance.....	14
3.4.2 Deciding Parameter of K.....	15

3.4.3 Pseudo-Code Implementation of k-NN.....	16
CHAPTER 4: PROPOSED MODEL FOR PREDICTION.....	18- 32
4.1 Proposed Model.....	18
4.1.1 Neural Network.....	18
4.2 Implementation.....	22
4.2.1 Data Collection.....	24
4.2.2 Data Ranging and Classification.....	27
4.2.3 Predicting Location.....	29
4.2.4 Data Retrieving form External Server.....	30
CHAPTER 5: EXPERIMENTAL RESULT ANALYSIS.....	33-51
5.1 PLG.....	33
5.2 Logistic Regression.....	35
5.2.1 Accuracy of the Testing Datasets.....	36
5.2.2 Comparison with PLG.....	39
5.3 Decision Tree.....	41
5.3.1 Comparison with PLG.....	42
5.4 k-NN.....	44
5.4.1 k-NN Parameters.....	44
5.4.2 Comparison with PLG.....	46
5.5 Comparisons between the algorithms.....	49
CHAPTER 6 : CONCLUSION.....	52
6.1 Conclusion Remarks.....	52
6.2 Future Works.....	52
REFERENCE.....	53

List of Figures

Figure 3.1: Work flow of the project.....	7
Figure 3.2: Logistic Regression representation.....	8
Figure 3.3: Sample Training Dataset of Latitude and Longitudes.....	9
Figure 3.4: Plotting of selected (by predict function) Latitude and Longitude (places) in the graph.....	11
Figure 3.5: k-NN algorithm distance calculation.....	15
Figure 3.6: k-NN selecting parameter of K	16
Figure 4.1: Feed-Forward Neural Network.....	18
Figure 4.2: Implementation of Sigmoid function.....	20
Figure 4.3: Reason for switching to mobile's internal Database from Firebase's external Database.....	25
Figure 4.4: Shifting phase from Firebase Database to Internal Database.....	26
Figure 4.5: Data Ranging and Classifying for 'Unique Location List' and 'Time Based Unique Location Holder'.....	27
Figure 4.6: Range Limit Selection and Calculation Concept.....	28
Figure 4.7: Coordinates of Predicted Location and Their Accuracy.....	30
Figure 4.8: Data retrieving flow.....	31
Figure 4.9: Developed Mobile Application.....	32
Figure 5.1: Accuracy of PLG on Training & Testing Datasets (Graph).....	33
Figure 5.2: Accuracy of PLG on Training & Testing Datasets (Comparison Chart).....	34
Figure 5.3: Accuracy computation for the training dataset	35

Figure 5.4: Sample testing dataset.....	35
Figure 5.5: Accuracy of the 1 st testing dataset.....	36
Figure 5.6: Accuracy of the 2 nd testing dataset.....	36
Figure 5.7: Accuracy of the 3 rd training dataset.....	37
Figure 5.8: Accuracy of Logistic Regression on Training & Testing Datasets (Graph).....	38
Figure 5.9: Accuracy of Logistic Regression on Training & Testing Datasets (Comparison Bar Chart).....	38
Figure 5.10: Comparison between Logistic Regression & PLG (Accuracy %).....	40
Figure 5.11: Comparison between Logistic Regression & PLG (Accuracy %).....	40
Figure 5.12: Accuracy of Decision Tree on Training & Testing Datasets (Graph).....	41
Figure 5.13: Accuracy of Decision Tree on Training & Testing Datasets (Comparison Bar Chart).....	42
Figure 5.14: Comparison between Decision Tree & PLG (Accuracy %).....	43
Figure 5.15: Comparison between Decision Tree & PLG (Accuracy %).....	43
Figure 5.16: Training Accuracy.....	45
Figure 5.17: Validation Accuracy.....	45
Figure 5.18: Test Accuracy.....	46
Figure 5.19: Bar Diagram of accuracy for training and testing datasets.....	47
Figure 5.20: Line Diagram of accuracy for training and testing datasets.....	47
Figure 5.21: Comparison between k-NN & PLG (Accuracy %).....	48
Figure 5.22: Comparison between k-NN & PLG (Accuracy %).....	49
Figure 5.23: Comparisons between algorithms (accuracy).....	50
Figure 5.24: Comparisons between algorithms (accuracy).....	50

List of Tables

1. Table 1: Stored Location Data Format-----	26
2. Table 2: Time Based Unique Prediction Location History List: 6 pm-----	30
3. Table 3: Accuracy for the datasets-----	33
4. Table 4: Accuracy for the datasets-----	37
5. Table 5: Comparison with PLG(Accuracy %)-	39
6. Table 6: Accuracy for the datasets-----	41
7. Table 7: Comparison with PLG (Accuracy %)-	42
8. Table 8: Accuracy for training and testing datasets-----	46
9. Table 9: Comparison with PLG (Accuracy %)-	48

CHAPTER 1

INTRODUCTION

This chapter contains Motivation, Objectives & Goals and Thesis orientation which will introduce our thesis project. This chapter will give an overview of the thesis topic and other related information.

1.1 Motivation:

In our day to life we roam around many places. For that, we need much kind of preparations. For example, on Friday someone always goes to a park on 4:00 PM for that he needs to prepare shoes, dresses etc. What if, we make a system where the system will tell the user where will be his next location that means it will predict the location of the user and suggest all the necessary gadgets, gears, all the visiting places near it and traffic updates to reach that place. Obviously, it will be a very useful system to use in our day to day life. Moreover, every user can know their next location for that he can manage his routine of daily life one step ahead. After that, people can be aware of all the traffic updates which will give the user a better way for choosing the transportation. In result, valuable time can be saved. While researching these factors we get motivated to build a system where every people can be benefited from our system having the prediction of their next location and suggestion of traffic updates and important places user can visit. Thus, we motivated to use the Machine Learning Algorithm for the prediction of the user's location using their previous visited location.

1.2 Objectives & Goals

This thesis mainly focuses on the predicting the next possible location of the person by analyzing his past locations and current location. The proposed algorithm “Predictive Location Generation (PLG)” uses the GPS data and machine learning techniques to predict next possible location. Machine learning algorithms such as Decision tree, KNN, Logistic Regression etc can be trained by providing training datasets to them and then these algorithms can predict the data by comparing the provided data with the training datasets. Our objective is to train our algorithm

(PLG) by providing training datasets to it and our goal is to generate the next possible location of a person by using PLG.

1.3 Thesis Orientation

Chapter 1 is the **INTRODUCTION** of the thesis. The Motivation and Objectives & Goals of the thesis are described here.

Chapter 2 is **LITERATURE REVIEW**. This chapter consists of “Literature Review” which indicates our information collection repository. This chapter also consists of “Related Works and Research” which indicates to the real life works and researches done by others, which are related to our thesis work in many ways.

Chapter 3 is **METHODOLOGY** where three algorithms of machine learning are described and be shown their implementation for comparing them with our proposed model.

Chapter 4 is **PROPOSED MODEL FOR PREDICTION**. This chapter consists of “Proposed Model” and “Implementation”. “Proposed Model” introduces our developed model, its works, and its features, its workflows through description, flowcharts, and block diagrams and so on. “Implementation” section describes how to implement our proposed model and how to get the predict locations by using our model.

Chapter 5 is **EXPERIMENTAL RESULT ANALYSIS** where we have shown the comparisons between the three machine learning algorithms and our proposed model in terms of accuracy of the predicted locations. We have shown here how our proposed model is giving better prediction then those three machine learning algorithms by analyzing the found results and accuracy. For analyzing, we have used tables, different kind of diagrams, and different kind of comparison graphs and so on.

Chapter 6 is **CONCLUSION** which consists of “Conclusion Remarks” and “Future Works”.

CHAPTER 2

LITERATURE REVIEW

This chapter contains literature review related with Predictive Location Generation (PLG), Logistic Regression, Decision Tree, k-NN. This chapter also refers the related works and research. Besides, this chapter will also give information about our research activity.

2.1 PLG

In PLG (Predictive Location Generation) we have applied Neural Networking approach. Firstly, debugging a learning algorithm then machine learning diagnostics has implemented. After that, we have evaluated a hypothesis which made our algorithm worked better then we selected the model and trained validation test datasets. Moreover, we have diagnosed our datasets by regularization. Next, we have learned the curves so that, we could classify the data cluster and made the prediction [19]. According to A Medium Corporation, among three layers of Neural Network output layer is very important for predicting any value. For that, we have needed a threshold value and if this value is greater or lesser then an output produced of 1 or 0 respectively. After that, sigmoid function has deduced the output result where the result only covered the answer “maybe” rather “yes/no”. This function is necessary for the network’s learning capabilities [20]. This PLG model, we have used bias value here for computing the hidden layer value [21]. In addition, this model has used Feedforward Neural Network and the goal for the network is to determine some approximate functions and learned the value which gave the best function result [22]. Next, this model has used list of cost functions where we could measure the goodness of the model [23]. In addition, we have used the Feedforward Backpropagation Neural Network which enabled the field of pattern recognition to one step ahead [24]. Next, we have used Backpropagation which is the primary concern of learning in neural network [25]. This model is supervised learning because the training dataset worked as a teacher supervising the learning technique [26]. In contrast, of conventional serial computer the neural network have used many simple central processors which take the weighted sum of their inputs from other processors [27].

2.2 Logistic Regression

For comparison of the accuracy of PLG, we have used Logistic Regression, Decision Tree, k-NN. Like all regression algorithm, the Logistic Regression is a prescient investigation. Logistic Regression is utilized to depict information and to clarify the connection between one ward twofold factor and at least one ostensible, ordinal, interim or proportion level autonomous factors. The logistic regression calculates the log odds of an event [5]. However, we need to keep in mind that, adding too many values to the dataset of training for the logistic algorithm makes the algorithm inefficient and creates over fitting problem [6]. So, we are careful about this factor while implementing and training the logistic regression algorithm for comparing the accuracy. The Logistic Regression computes the cost and gradient vector. In this process, the logistic regression is using “sigmoid function” [7][10]. Then using the results found from Sigmoid function, the algorithm computes the cost and the gradient [7][9][11]. Then algorithm is using a optimization and accuracy measuring function for the final, optimal result and accuracy [9][10][11].

2.3 Decision Tree

Decision Tree is a machine learning algorithm which takes the decision based on the result of certain conditions checking and testing [13]. No specially processed data is needed for using in the decision tree, because the decision tree itself can classify data and based on this classification, the tree performs regression, clustering, and computing probability and so on [12].The tree’s non leaf nodes have input attributes and leaf nodes have classification factors [12][13]. Entropy function is used in the decision tree algorithm to characterize the impurity of a single dataset [12][13]. Besides, information gain and remainder functions are so much helpful for the decision tree algorithm [12][13].

2.4 k-NN

K-NN mainly uses Euclidean distance between new data and training data points [14][15][16]. The found distances are needed to be sorted in decreasing order [15][16]. Next, assuming the number of the neighbor points is positive; this positive number is needed to be filtered from the sorted list [16][18]. A small value of neighbor’s number has greater influence on the result

[14][18]. Large number of neighbor points increases the cost of the algorithm and decrease the efficiency [14][18]. In this kind of situation, cross validation technique is very useful to increase the efficiency of the outputs [11][16][17].

2.5 Related Works and Research

“Hierarchical Destination Prediction Based on GPS History” is research work done by Wenhong Huang, Man Li, Weisong Hu, Goujie Song, Kunqing Xie in 2013 [1]. They have proposed a location activity- mapping method and employed Hidden Markov Model for hierarchical destination prediction and they proposed to this in a supervised way. They have validated their approach by using the GPS data of 100 real world users. They stated that, their accuracy could be improved by 3% - 8%.

Binh T. Nguyen, Nhan V. Nguyen, Nam T. Nguyen and My Huynh T Tran has proposed model for predicting location from historical movement [2]. They have used mobile GPS for collecting the locations of the person and they have developed an android application to collect and study the user location. They have compared their proposed model’s result with three different supervised machine learning approaches: SVM, decision tree and Markov model. They have got the highest accuracy 92% for the SVM model.

In “Learning Significant Locations and Predicting User Movement with GPS” research paper, D. Ashbrook and T. Starner has proposed a probabilistic model of destination prediction using GPS data [3]. They have applied a Markov model to find the highest probabilistic location being visited next from a dataset of recently visited locations.

In “Can your Friends Predict Where You will be?” research work, Lei Cao and James She has proposed a model to predict the user’s future locations based evaluating the user’s regular mobility patterns and social influence[4]. They have applied Markov model for this research.

“Deepsense: A Novel Learning Mechanism for Traffic Prediction with Taxi GPS Traces” is research work done by Xiaoguang Niu, Ying Zhu and Xinging Zhang [8]. In this research work, they have proposed DeepSense mechanism and a trained temporal-spatial deep learning algorithm to collect information from the high dimensional, nonlinear and random traffic flow for predicting good results.

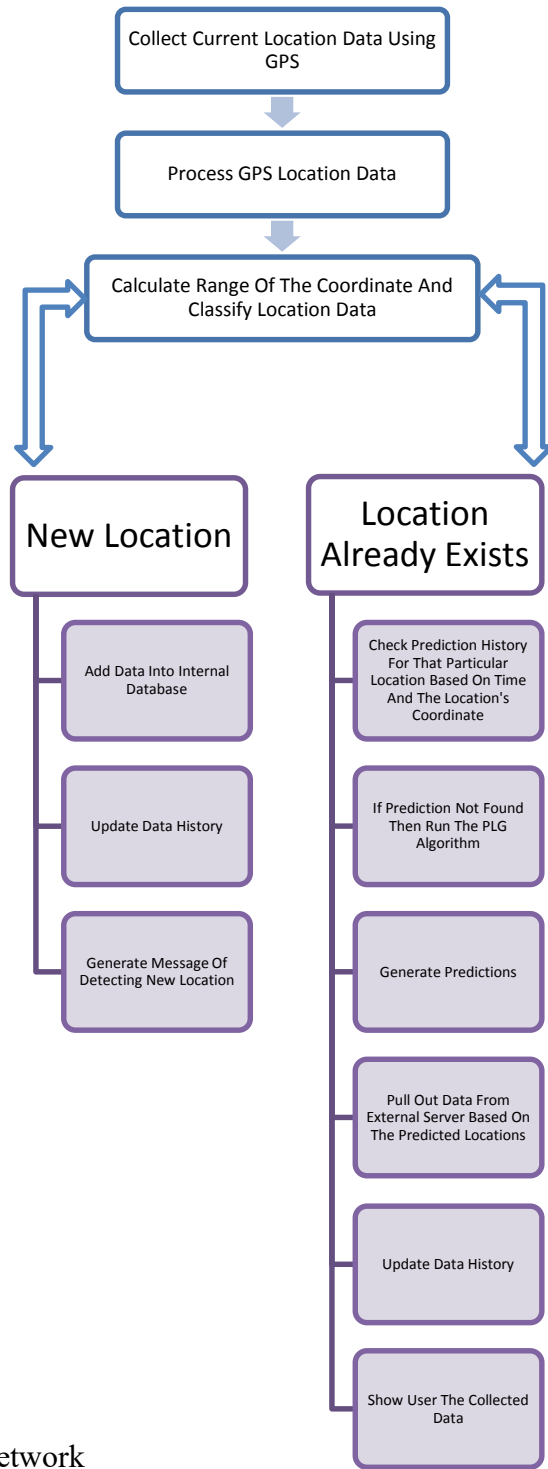
CHAPTER 3

METHODOLOGY

This chapter contains the methodology of Predictive Location Generation Using Machine Learning Approach, Logistic Regression, Decision Tree and k-NN algorithms. This chapter will describe how the following algorithms work.

3.1 Predictive Location Generation Using Machine Learning Approach

Our target is to generate predictive future locations based on user's current location and then based on the predicted locations produced our next task is to retrieve data from external servers. External servers will provide us with data about crime report, traffic reports, and new product launch advertisement and so on. To accomplish the project our course of work includes collecting data dynamically through mobile phone's GPS system. Then process the location data to organize data into time, coordinate and day format. After that, we will try to choose a suitable range calculation and classification method for location coordinates to identify the positions and uniqueness of the coordinates. Based on the result of range calculation and classification, we will try to either generate predictions or just update database for new locations. For generating predictions based on current location, we need to correlate visited values (numerical) and distances of the coordinates stored in the mobile's database. Now, we will use Neural Network Approach among the machine learning algorithms to predict the next possible locations of the user using the stored coordinates in the mobile's internal database. To reduce error of the output generated by system, we will use improved Back Propagation technique.



PLG: Predictive Location
Generator using Neural Network

Figure 3.1: Work flow of the project

3.2 Logistic Regression

Logistic Regression is the suitable relapse examination to lead when the reliant variable is dichotomous (paired). Like all regression algorithm, the Logistic Regression is a prescient investigation. Logistic Regression is utilized to depict information and to clarify the connection between one ward twofold factor and at least one ostensible, ordinal, interim or proportion level autonomous factors. At the center of the logistic regression analysis is the task estimating the log odds of an event. Mathematically, logistic regression estimates a multiple linear regression function defined as:

$$\text{logit}(p) = \log\left(\frac{p(y=1)}{1-p(y=1)}\right) = \beta_0 + \beta_1 x_a + \beta_2 x_a + \dots + \beta_r x_m$$

for $i = 1 \dots n$.

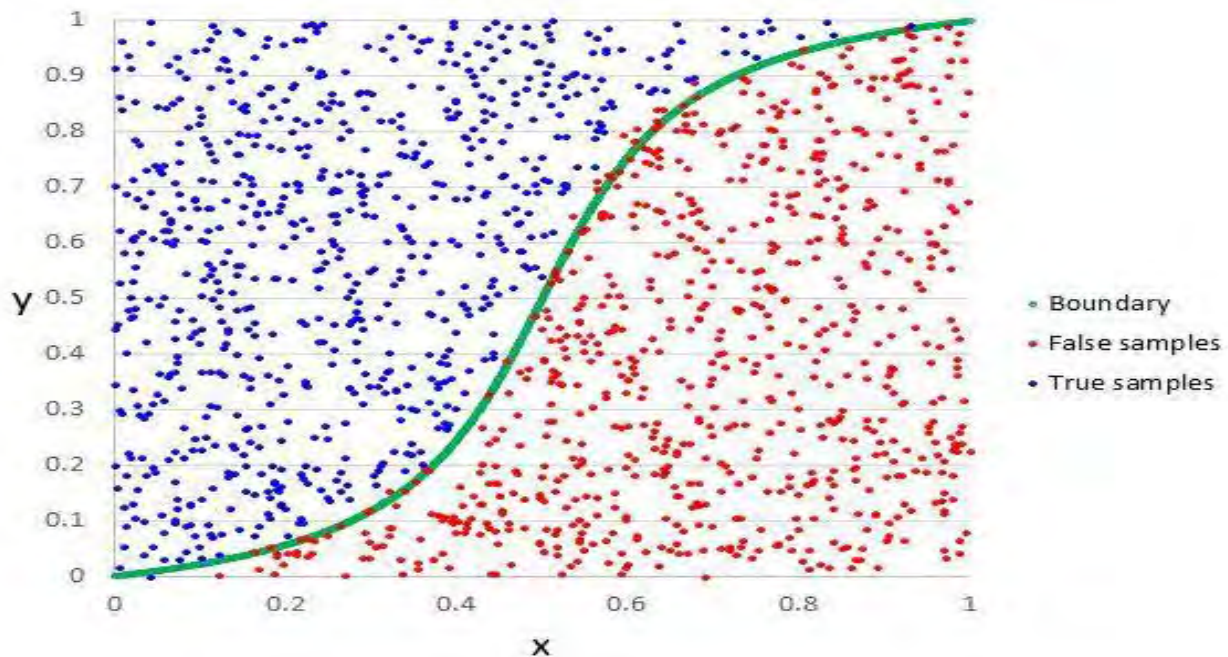


Figure 3.2: Logistic Regression representation

However, independent variables adding to a logistic regression model always enhance its statistical validity. But, adding more values makes the model inefficient and occurs over fitting

Next, the Logistic Regression algorithm computes the Cost and Gradient. We initialized the Theta value to zero (0) and the correct values are determined by the optimization algorithm later. Here, we need (n+1) theta values ($\Theta_0, \Theta_1, \Theta_2, \dots, \Theta_n$).

Now, The Logistic Regression algorithm has a function “[J, grad] = computeCost(initialTheta, X, Y)” which is called “computeCost.m”. This function will determine the cost and gradient vector.

In this function, there is a variable named “m”. “m” is the number of training examples. An ones column is added to the “X” that represents the bias term. This function computes the hypothesis using the following formula:

$$h_{\Theta}(x) = g(\Theta'X);$$

$$g(z) = 1 / (1 + e^{-z})$$

Here, g is called the sigmoid function and we have implemented it separately in another function called “sigmoid.m”.

$$h = \text{sigmoid}(X * \Theta);$$

$$\text{function } g = \text{sigmoid}(z)$$

Now, “computeCost” function computes the cost using the following formula:

$$J(\Theta) = -1/m [\sum y \log(h_{\Theta}(x)) + (1 - y) \log(1 - h_{\Theta}(x))]$$

Next, the gradient is being computed by using the following formula:

$$\text{Grad}(i) = 1/m \sum [(h_{\Theta}(x_i) - y_i) x_{ij}]$$

Next, we have called an optimization algorithm, which gives the correct values of the Theta that will minimize the cost function.

$$\text{options} = \text{optimset}('GradObj', 'on', 'MaxIter', 400);$$

$\theta = \text{fminunc}(@(\text{t})\text{computeCost}(\text{t}, \text{X}, \text{Y}), \text{initialTheta}, \text{options});$

Now, we have created a function “predict.m” to compute the predictions at first, which takes the values of theta and X as parameters and compute the prediction by calling sigmoid function. This function predicts or selects some points (Latitudes and Longitudes).

Now, we plot the correct and selected (by predict function) values of the training dataset, which gives the graph of selected points. These points represent the predicted places for our training dataset.

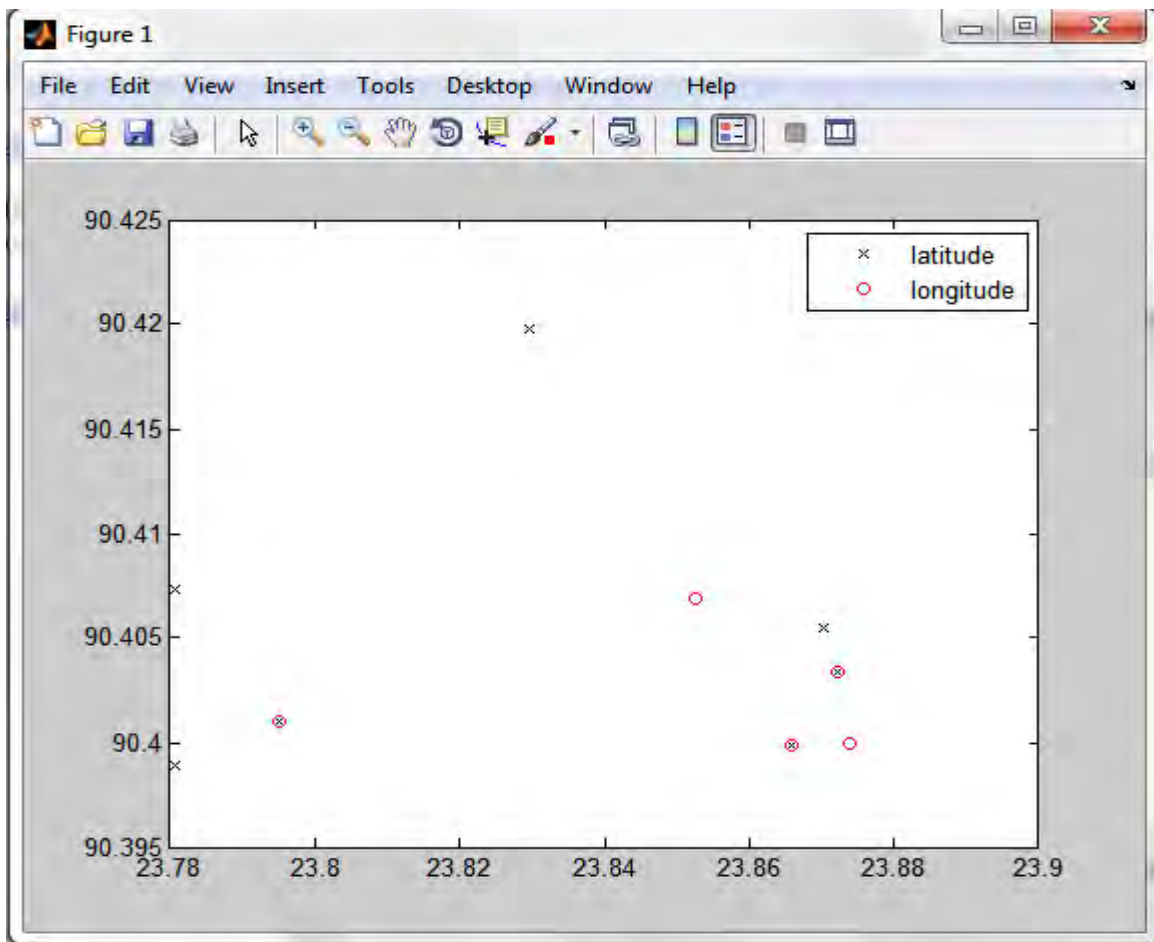


Figure 3.4: Plotting of selected (by predict function) Latitude and Longitude (places) in the graph.

Now, we have computed the percentage of accuracy by calling the “mean” function.

$\text{accuracy} = \text{mean}(\text{double}(\text{predictions} == \text{Y}) + 100)$

3.3 Decision Tree

A decision tree is a tree used for machine learning where it takes decision based on the outputs of a set of condition checking and testing. Its each of the non-leaf has an input attribute (A_i) and leaf node has a positive (+) or negative (-) classification. Each of the edges of the tree is assigned with one of the possible values of the attribute.

For characterizing the impurity of a dataset, decision tree uses the entropy function.

$$\text{entropy}(\text{dataset}) = -(p_+ * \log_2(p_+) + p_- * \log_2(p_-))$$

Now, the Information Gain for the decision tree is

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{remainder}(A)$$

Here,

$$\text{remainder}(A) = \sum_{i=1}^v \frac{p_i + n_i}{p+n} I\left(\frac{p_i}{p+n}, \frac{n_i}{p+n}\right)$$

3.3.1 Pseudo-Code Implementation of Decision Tree

GrowTree(D, A) // tree with attribute for the training dataset

Input: data D ; set the attributes A

Output: attribute tree T associated with labeled leaves.

If Homogeneous(D)

return label(D);

$S = \text{BestSplit}(D, F);$

Split D into substes D_i ;

If D_i is not empty

$T_i = \text{GrowTree}(D_i, F);$

else

T_i is a leaf labeled with Label(D);

return tree; // root labeled with S and children are T_i

Bestsplit(D,F) // best split for the decision tree

Input: data D; //set of attributes

Output: attributes A to split

I_{min} = 1;

for each f ∈ F do

split D into subsets D₁; // D₁ based on the values of f

if impurity({D₁, ..., D_n}) < I_{min} then

I_{min} = Impurity({D₁, ..., D_n});

f_{best} = f;

end

end

return f_{best};

3.4 k-NN

k-NN is an algorithm for finding pattern or pattern recognition. It is used for classification and regression. This method is very popular in machine learning technique. Moreover, this method needs characteristics of observations which are collected for both training and test dataset. In k-NN approach, there are two important concepts, first one is the method of calculate the distance between two variables (any characteristics of observations). Secondly, we should know the parameter *K* which is the number of neighbors that will be chosen for k-NN approach.

3.4.1 Calculate Distance:

There are many methods we can follow to calculate the distances between the new data point and all the training data points. But by default, the k-NN() function uses Euclidean distance which can be computed with this equation:

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

Where x and y are to be compared with k characteristics. If we break it down then we will have the numerical point of the distance between the x and y point. For example, there are two points $P1$ and $P2$ where,

$$P1 = (w1, x1, y1, z1) \dots \dots \dots (1)$$

$$P2 = (w2, x2, y2, z2) \dots \dots \dots (2)$$

From (1) & (2)

$$\text{Euclidian distance } (p1, p2) = \sqrt{((w1 - w2)^2 + (x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2)}$$

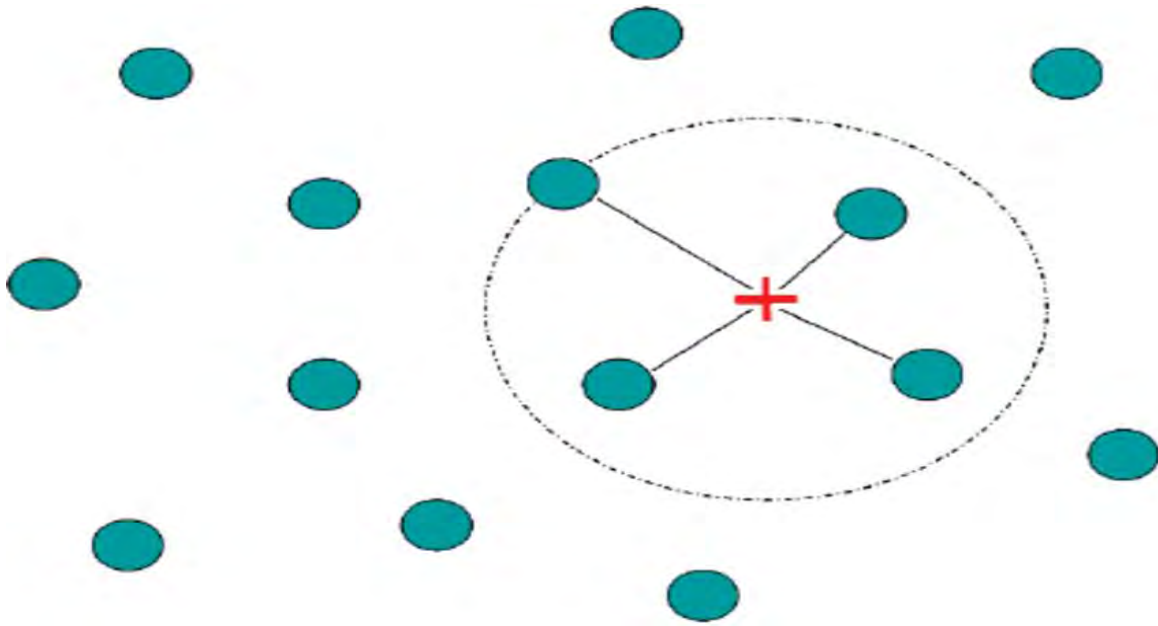


Figure 3.5: k-NN algorithm distance calculation

3.4.2 Deciding Parameter of K:

Next, arrange the distances in non-decreasing order. The number of neighbors which K represents can be based on the dataset. It can vary for different set of problems. After that, we assume that K is a positive value. Moreover, filtering K values will take place from the sorted list. Thus, we get the K number of distances. The choosing the value of K is very important. A small value of K represents the noise will have greater influence on the result. If we choose a large value of K , the computational expense increases and de-motivates the raw idea of k-NN. In result, a simple way to choose K is:

$$K = n^{(\frac{1}{2})}$$

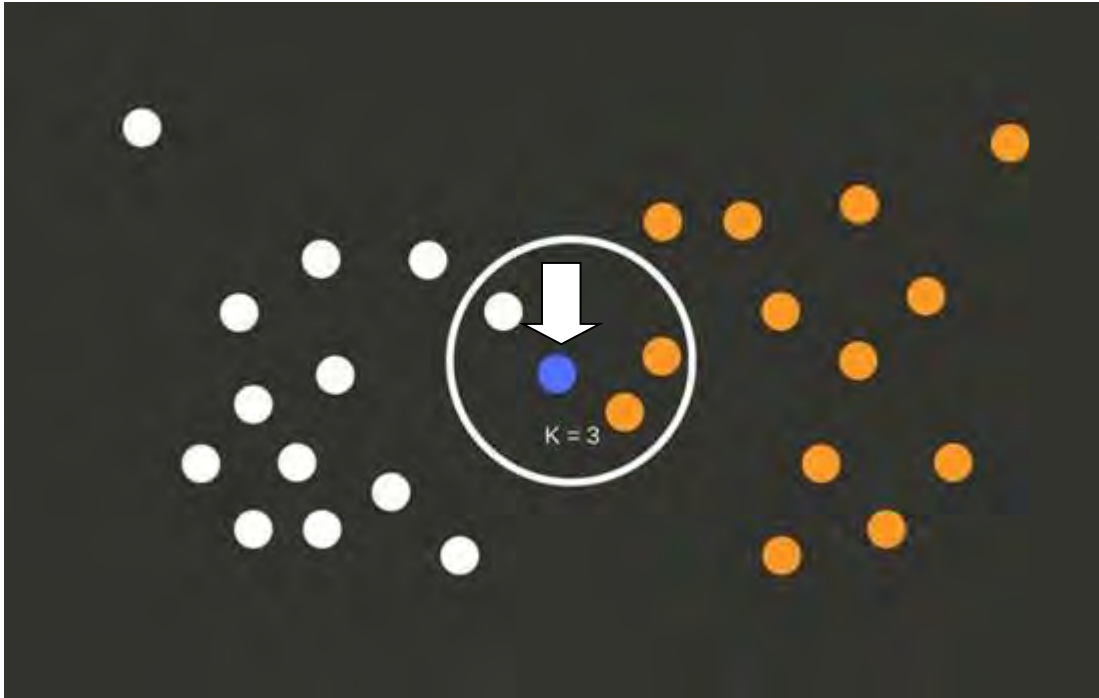


Figure 3.6: k-NN selecting parameter of K

If we use cross validation, we might increase efficiency of the results. In addition, cross validation allows us to test k-NN algorithm with different values of K . In this figure, there are two different classes for the arrowed circle. Where the K value is 3, we calculate the similarity distance by implementing Euclidian distance. If the similarity score is less that means the circles is near and creates a region of 3 circles around the arrowed circle. In the neighborhood, on the right side of arrowed circle has 2 circles of one class and on the left side there are 1 circle of another class.

3.4.3 Pseudo-Code Implementation of k-NN:

k- Nearest Neighbor

Classify (X, Y, x)

// X : training data of our GPS co-ordinates, Y : class labels of X i.e. which GPS co-ordinate represents which class, x : unknown sample i.e. it can be any GPS co-ordinate data

for $i = 1$ **to** n **do**

Compute distance $d(X_i, x)$

// training data and sample data distance computed using Euclidian method

End for

Compute set I containing indices for the k smallest distance $d(\mathbf{X}_i, x)$

// finding the indices of 'k' number means Euclidian Distances

Return majority class label for $\{Y_i$ where $i \in I$

CHAPTER 4

PROPOSED MODEL FOR PREDICTION

This chapter is about our proposed model for prediction. This introduces with Neural Network and how to implement the algorithm to predict the next possible location of the user.

4.1 Proposed Model

4.1.1 Neural Network:

This is a learning model which is notable for being adaptive in nature. Every node of Neural Network has their own sphere of knowledge about rules and functionalities to develop it-self through experiences learned from previous training. In a nutshell, it can adjust itself to the changing environment as it learns from initial training and subsequent runs provide more information about the world. In our project ‘learning’ process is a supervised process. Our neural network model is centered on weighting the predefined and already classified input streams, which is how every node weights the importance of input from each of its predecessors. Initially inputs are weighted arbitrarily but after starting of learning data and adjusting the error, inputs that help to get more right answers are weighted higher.

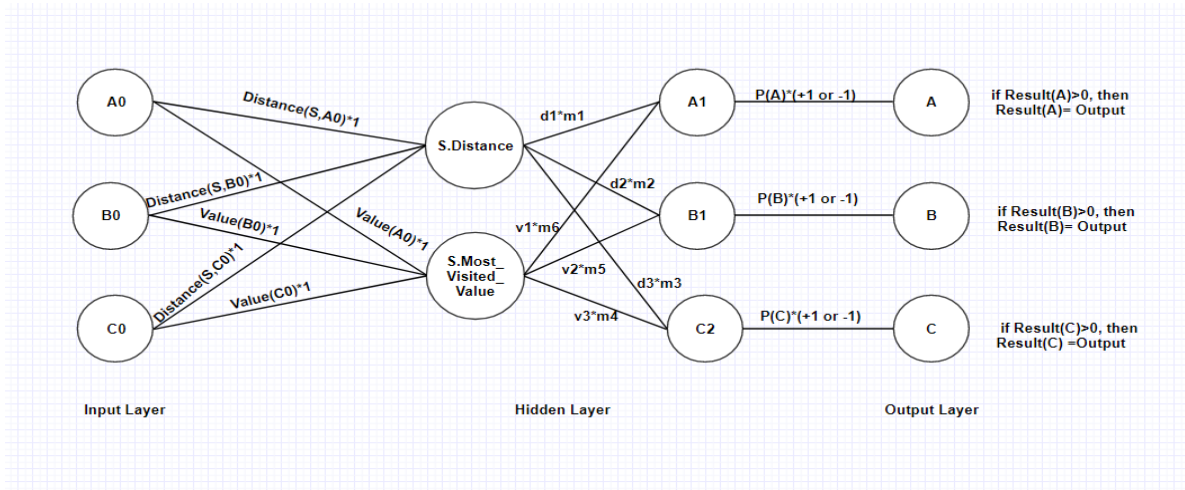


Figure 4.1: Feed-Forward Neural Network

This neural network model will take only one input at a time and will not allow multi-threading. Above figure consists of total three inputs, A0, B0 and C0 but these input locations cannot be able to use the model at the same time. In this model inputs A0, B0, C0, locations are most probable future locations (future locations are identified using range calculation, time base sorting and classification method) of the current location S which is detected dynamically using GPS.

Every possible future location Node (A0, B0 or C0) will hold two data within its node. One is distance between 'S' to that particular input future location. Another one is value of the total appearance of the input location. Here 'S' is the current location.

- Input Layer = this layer takes one input at a time.
- Hidden Layer = this layer has two additional sub-layer.
- Output Layer=this layer produces outputs.
- S = Source Node = Current Location Coordinate.
- Neural network's Input = X = A0, B0 or C0.
- Distance(S, X) = Distance between Location S and Location X
- Value(X) = total number of visit to the location X in a particular time of a day.
- P = priority weight which initially sets into 1.
- S.Distance = (Neuron node in hidden layer that hold the distance value from S to X) * (+P) +1.
- S. Distance = S. X.
- S.Most_Visited_Value = (Total number of visits to Location X) * (+P) +1 = S. V.
- S. Most. Visited_Value = S. V.

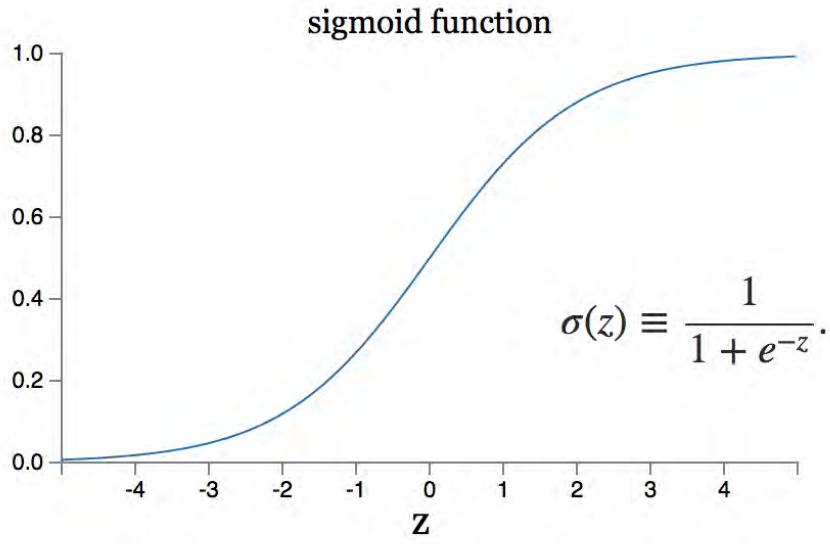


Figure 4.2: Implementation of Sigmoid function

In the project Predictive Location Generation Using Machine Learning, we will use sigmoid function for calculating probability of the future locations. We know that sigmoidal units will give high output when input is very positive and will give a low output when input is very negative. Besides this, sigmoidal units are highly sensitive to their input when input is near to zero. This last property of sigmoid function will help us to relate two different properties (distance and total visited number in the particular node) of any input. The properties of sigmoid function eventually will help the Neural Network model to generate desired results.

- $d = \frac{1}{1 + e^{-(-S.X)}}$ [where, $d=d1, d2, d3 \dots dn$]
- $v = \frac{1}{1 + e^{-S.V}}$ [where, $v=v1, v2, v3 \dots vn$]
- $Y = A1, B1 \text{ or } C1.$

- $m =$ Arbitrary weight from range zero to positive one, assigned randomly by the machine. $[m=m_1, m_2, m_3 \dots m_n]$

- $Y = (d * m) + I + (v * m) + I$

- $P(Y) = \frac{1}{1 + e^{-Y}}$

- Output = Positive values generate from Y.

4.1.2 Error Calculation:

$$Total_j = \sum w_{ij}x_j + \theta_j \dots \dots \dots (1)$$

$Total_j$ = The weighted sum of the input to the j^{th} node in the hidden layer when specified training pattern is given to input layer.

θ_j = weighted value from a bias node = 1.

$$A_j = x_z = \frac{1}{1 + e^{-Total_j}} (2)$$

A_j is differentiable activation function formulated from the Sigmoid equation.

$$\Delta_Z = E_Z - A_Z \dots \dots \dots (3)$$

Z = Output node Z.

E_Z = Expected target output for node for Z

A_k = Activation value of output node Z.

The error signal for node Z in the output layer can be calculated as -

$$\delta_Z = (E_Z - A_Z)A_Z (1 - A_Z) \dots \dots \dots (4)$$

$A_Z(1 - A_Z)$ = derivative of the Sigmoid function.

$$\Delta w_{j,z} = I_r \delta_z x_z \dots\dots\dots(5)$$

$$w_{j,z} = w_{j,z} + \Delta w_{j,z} \dots\dots\dots(6)$$

Above formulas are used to adjust the weight, $w_{j,z}$.

$$\Delta w_{j,z}^n = I_r \delta_z x_z + \Delta w_{j,z}^{n-1} * \mu \dots\dots\dots(7)$$

$\mu = momnetum$ = Value between 0 and 1.

A modification to the equation (5) is for the improvement of weight adjustment process.

$$\delta_z = (E_z - A_z)A_z(\sum w_{j,z}\delta_z) \dots\dots\dots(8)$$

δ_z = the error signal for node j in the hidden layer.

4.2 Implementation

PLG = Predictive Location Generation is feed forward neural network model based future location prediction software.

PLG divides its allocated database into 50 major segments where interrelated data will be stored for producing predictive locations. These segments have different purposes. First segment is denoted as ‘Unique Location List’, second segment is for detected ‘Location History’. Next 24 segments will be used for holding the unique locations, visited by users in every single hour. Let us, just call these segments as ‘Time Based Unique Location Holder’. Suppose segment ‘A’ will hold all unique visited places of user for time 6pm and similarly segments B, C and other 21 segment will also hold unique locations for time 7pm, 8 pm and so on . Last 24 segments are for holding prediction histories, these segments are also time based and will be very useful to reduce cost calculation and time complexity of the Neural Network.

Algorithm for PLG:

Start,

1. Check, if PLG 'Service' is enable or not.
 2. If (enable == false) then go to line 3 else go to line 6.
 3. Prompt a message and ask user to enable service
 4. If user enable service, then go to line 6 else go to line 5
 5. Remain idle for another hour and then go to line 1.
 6. Triggering GPS dynamically to collect current location data;
 7. Location Data = X coordinates.
 8. Check, if the 'X' coordinates is unique or not by matching X with each data stored in 'Unique Location List'.
 9. If (X is unique == false) then go to line 10 else go to line 24,
 10. Send data 'X' for range calculation.
 11. Distance = Range_Calculation_Haversine_Method (X, each data from 'Unique Location List)
 12. If (if Distance < predefined range) then go to line 13 else line 25,
 13. Then 'X' is to be in range of that particular. Suppose that Coordinate is 'Y'.
 14. Update 'Location History'.
 15. Update 'Time Based Unique Location Holder List'
 16. Check into 'Time Based Prediction History List' for data 'Y'.
 17. If no prediction found then go to line 18 else go to line 21.
 18. Call Neural Network based Prediction method with data 'Y'.
 19. Coordinate Z = Prediction_Method(Y).
 20. Update 'Time Based Prediction History List'.
 21. Call external server and necessary data for location data 'Z'.
 22. Show collected data to user.
 23. Return to line 5.
 24. Else (X is unique == true),
 25. Insert data 'X' to 'Unique Location List'.
 26. Update 'Time Based Unique Location Holder List'.
 27. Shows message "Location is unique."
 28. Return to line 5.
- End.

4.2.1 Data Collection

In our project we have used total four data sets. Among these datasets, one is training data set and other three are testing data sets. We have collected data in two phases. In the initial phase, we have built an android app named ‘Location Data Collector’ [Section-4.2] just to collect real time data of current locations detecting through mobile’s GPS. Data has been collected in hourly basis. All we need to do was to enable data collecting service option and then it automatically has started collecting data at every hour until the service has switched off. Location Data Collector app uses real time database of Firebase, a mobile and web application cross development platform. The app was acting properly but as per our project’s demand, our data must not be stored in external database. As all data must have to solely store in mobile’s internal database, so in second phase of our data collection, we have built our own android application which is able to collect current location data using GPS and store it into mobile’s internal database. We have named our new software ‘PLG’. PLG stands for Predictive Location Generation or Generator. The functionalities of PLG regarding data collection through GPS is as same as Location Data Collector application. Unlike Location Data Collector, PLG has full functionalities for ranging and classifying detected locations and for predicting future locations of users. To collect location data for training dataset we have collected data from eight people, who installed our app in their mobile phone. After collecting location data from the users, we have selected one data set to be our training dataset. For testing data sets, we have taken our own movement history from the PLG app’s history installed our mobile phones.

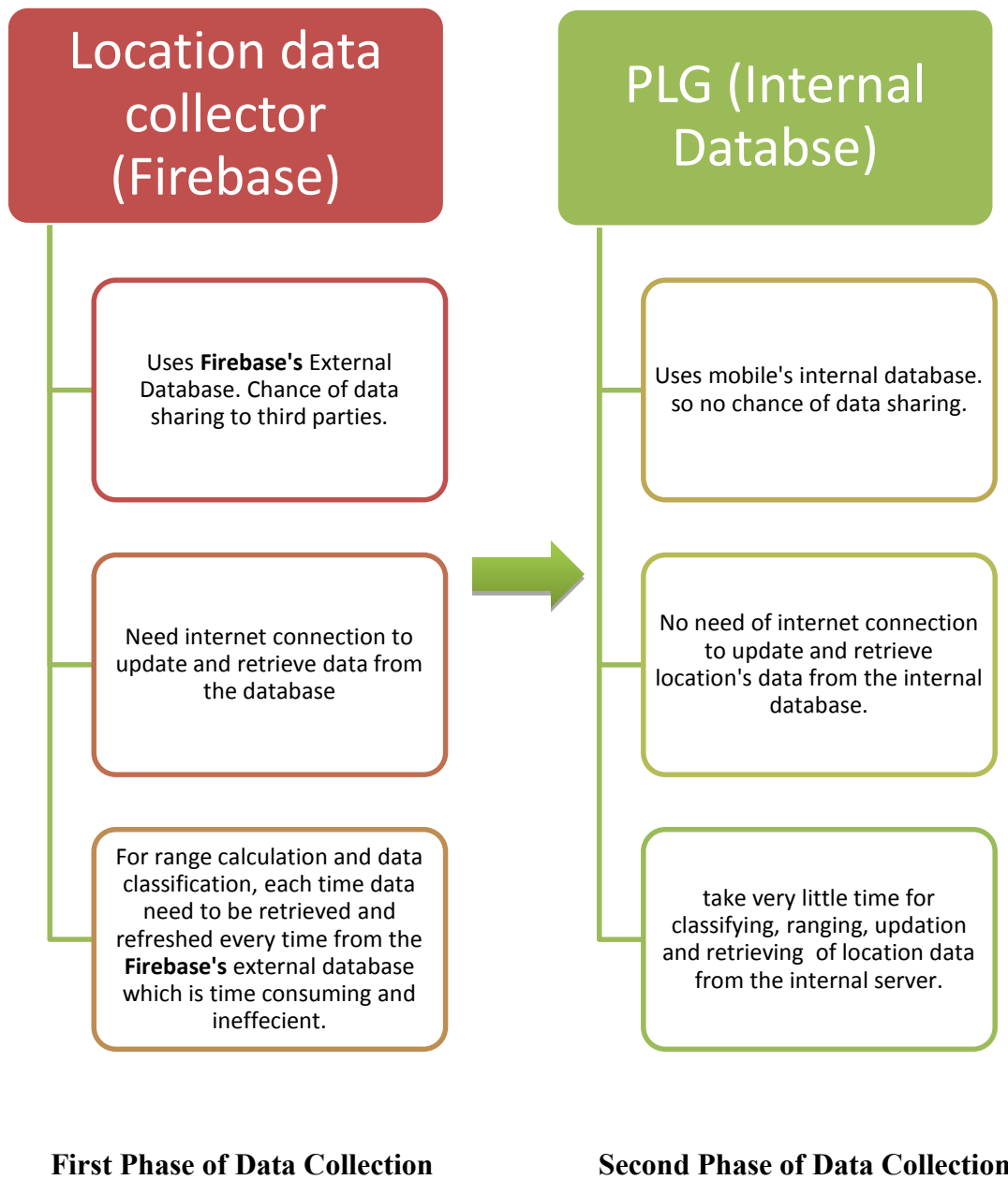


Figure 4.3: Reason for switching to mobile’s internal Database from Firebase’s external Database



Firestore's Database

Internal Database

Figure 4.4: Shifting phase from Firestore Database to Internal Database

[Hour - GPS Coordinate- Day] = [(0-23) hours – (Latitude, Longitude) – (Sunday to Saturday)]
 = [7 hours - 23.870263, 90.405461 – Saturday]

Table 1: Stored Location Data Format

Data Set	Number of Data
Training = Dataset 0	18144
Testing = Dataset 1	4320
Testing = Dataset 2	5760
Testing = Dataset 3	4680

4.2.2 Data Ranging and Classification:

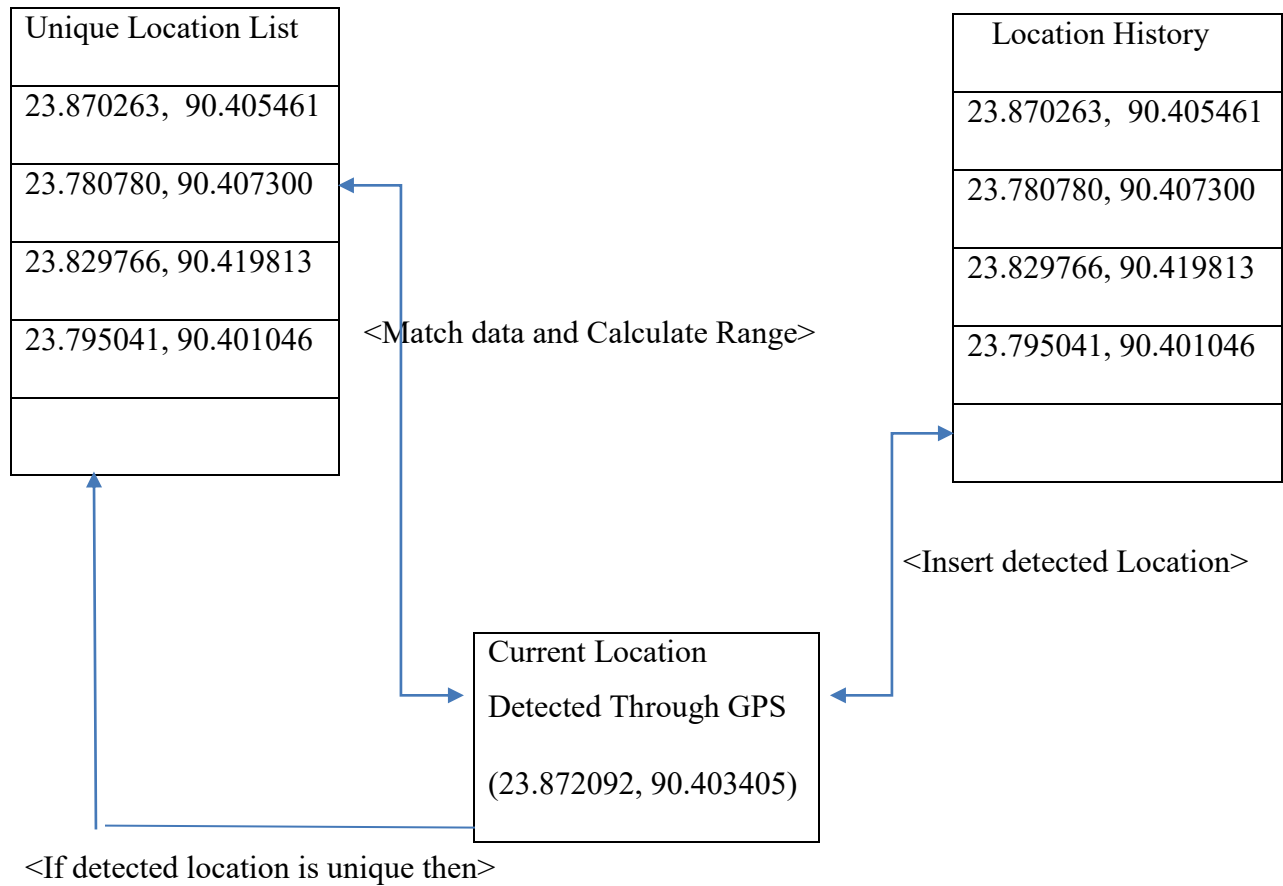


Figure 4.5: Data Ranging and Classifying for ‘Unique Location List’ and ‘Time Based Unique Location Holder’

When a location is detected by the GPS, PLG checks if the location is unique or not by matching the location with each locations from ‘Unique Location List’. This Unique Location List is a place in database where PLG stores all its unique places that are visited by a user. If the currently detected location does not match with any location of the list then PLG does range calculations using ‘Haversine Formula’ with each and every location data from the ‘Unique Location List’. If output of the range calculation is greater than predefined range limit (115m) then the location is unique, else location is not unique. If location is not unique then it must falls under the range of a location from the ‘Unique Location List’. We are doing range calculation

for two reasons. One is to identify the whole premises of a place another one is to reduce garbage location node from Neural Network's hidden layer to lessen the complexity of network's layers. To draw and calculate range our system uses 'Haversine' formula to calculate the great-circle between two points and then calculate the shortest distance between those two coordinates.

Haversine Formula for Range calculation and classifying data:

$$a = \sin^2(\Delta\phi/2) + \cos \phi_1 \cdot \cos \phi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d = R \cdot c$$

Φ = latitude.

λ = longitude.

R = mean radius of earth= 6371 Km.

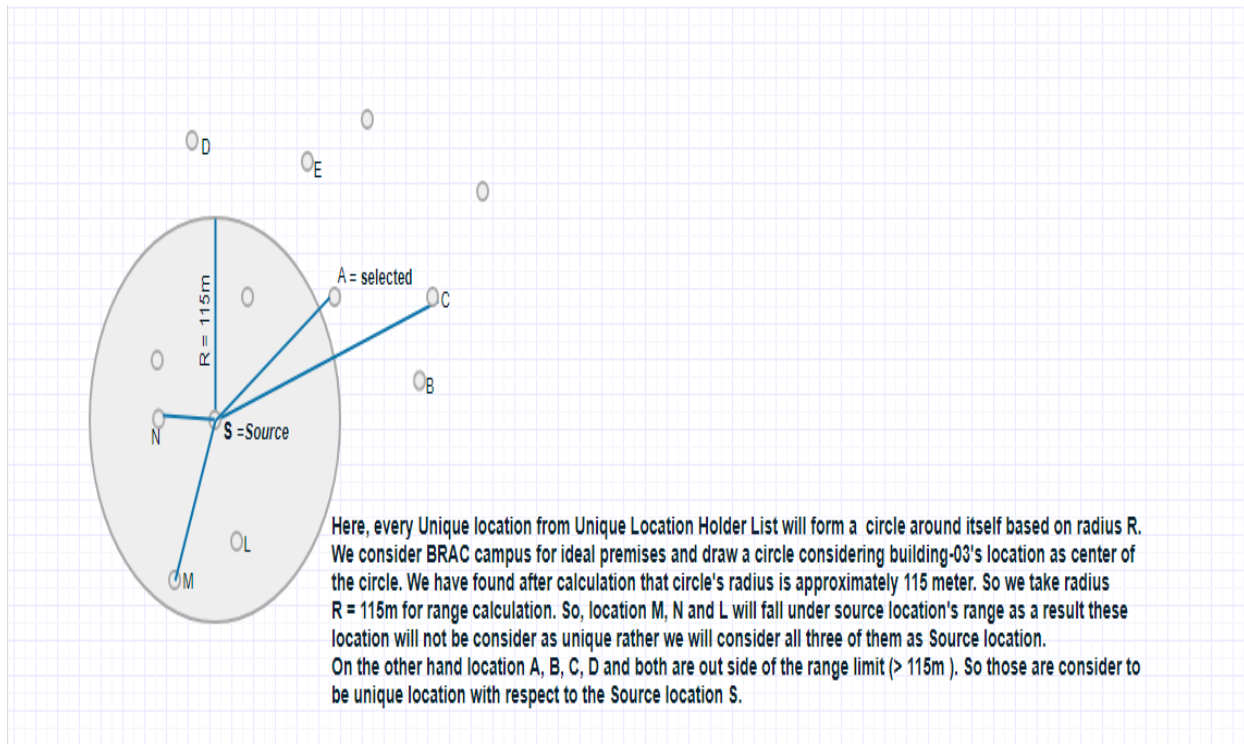


Figure 4.6: Range Limit Selection and Calculation Concept

After calculating range, inserting data into specific segments and linking data with ‘Unique Location List’ data, we just need to put each data on ‘Time Based Unique Location Holder List’. ‘PLG’ can easily do it as every current location data holding its detection time information within it along with its coordinate data information.

Current Location = [7 hours- 23.870263, 90.405461 – Saturday]

‘Time Based Unique Location Holder List’ will contain the total numbers of visits to a particular place visited by users. Besides this, ‘Time Based Unique Location Holder List’ will also contain Boolean values that will indicate if the location in a particular cell is new or old, means does user make any move from the stored location or not. The cells of the ‘Time Based Unique Location Holder List’ will look like this-

Time Based Unique Location Holder List: 6 Pm

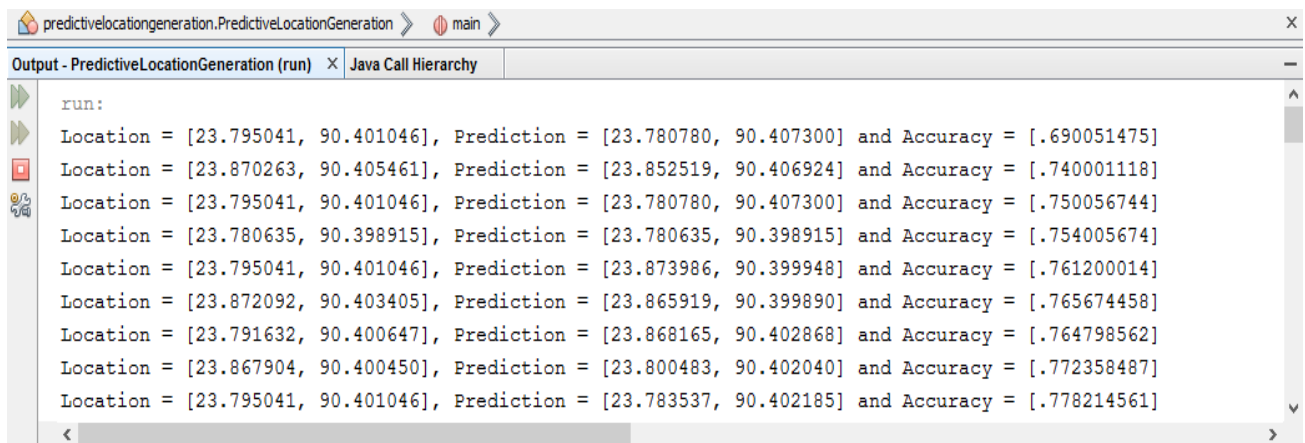
Location= b, total visit =518, movement = true	Location-a, total visit =221, movement = true	Location-f, total visit =18, movement = true	Location-d, total visit =308, movement = true	Location-c, total visit =1, movement = false
--	---	--	---	--

4.2.3 Predicting Location:

After range calculation and data classification, PLG will check ‘Time Based Prediction History List’, if prediction history is found for corresponding current location data then PLG will forward the data to data retrieving section for collecting data from external server. If ‘Time Based Prediction History List’ does not have any prediction regarding the current location coordinates then, current location will be forward to Neural Network based prediction method. This method will produce predictive location following Neural Network model’s methodology [Section 4.1]. After getting the predictive location coordinates PLG then, send this coordinates to data retrieving section.

Table 2: Time Based Unique Prediction Location History List: 6 pm

Location	Predicted Location
23.780780, 90.407300	23.780635, 90.398915
23.870263, 90.405461	23.873986, 90.399948
23.852519, 90.406924	23.795041, 90.401046



```
run:
Location = [23.795041, 90.401046], Prediction = [23.780780, 90.407300] and Accuracy = [.690051475]
Location = [23.870263, 90.405461], Prediction = [23.852519, 90.406924] and Accuracy = [.740001118]
Location = [23.795041, 90.401046], Prediction = [23.780780, 90.407300] and Accuracy = [.750056744]
Location = [23.780635, 90.398915], Prediction = [23.780635, 90.398915] and Accuracy = [.754005674]
Location = [23.795041, 90.401046], Prediction = [23.873986, 90.399948] and Accuracy = [.761200014]
Location = [23.872092, 90.403405], Prediction = [23.865919, 90.399890] and Accuracy = [.765674458]
Location = [23.791632, 90.400647], Prediction = [23.868165, 90.402868] and Accuracy = [.764798562]
Location = [23.867904, 90.400450], Prediction = [23.800483, 90.402040] and Accuracy = [.772358487]
Location = [23.795041, 90.401046], Prediction = [23.783537, 90.402185] and Accuracy = [.778214561]
```

Figure 4.7: Coordinates of Predicted Location and Their Accuracy

4.2.4 Data Retrieving from External Server:

This is the last section where PLG will use predicted location coordinates and send request to external server for information data like traffic updates, new product launch, and crime report and so on corresponding to the predicted location. After retrieving desired data from external server, PLG will show it to mobile phone's screen of the user. To implement this portion in PLG, we have faced several problems and the most difficult one is to retrieve data automatically from external server based on location coordinates. As our application, PLG is still in its experimental phase and there is no better solution (except for manual data collection from

external servers)to retrieve data automatically based on location coordinates, so we have planned to collaborate our project with another already implemented BRAC university project named “Crime Mapping Through Digital Data Analysis From Intermediate Repository by Crowd Sourcing”. Dr. Amitabha Chakrabarty has supervised this project. According to authors, Prema Dev, Samrin Sultana Tithi and Sabiha Islam Aunamika, their project will deploy a real time crime mapping system service in Bangladesh that will make a combination of crowd sourcing crime incidents from public and extract crime information from various social networking sites based on the geographical location of the crimes. Therefore, our plan is to provide them PLG generated geographical location automatically and extract data from their server. Currently we are working with only one external server but we have a plan to expand server numbers by collaborating our project with different organization’s data servers and to produce a better solution (technical) for resolving data retrieving problems.

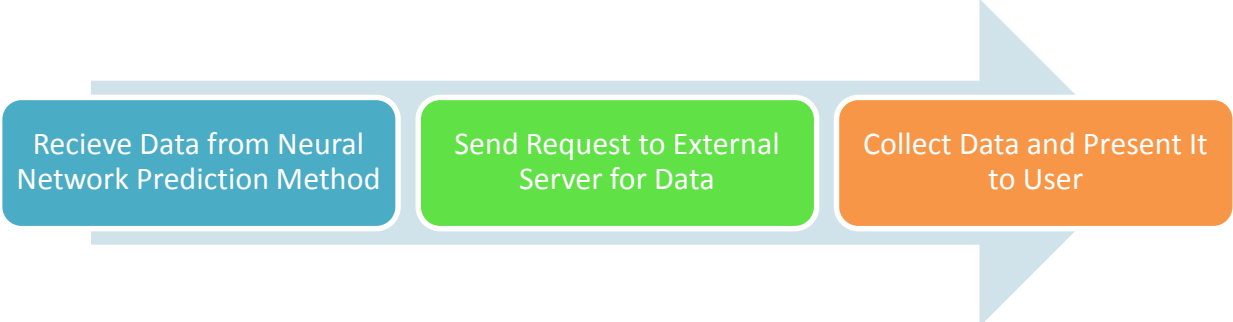


Figure 4.8: Data retrieving flow

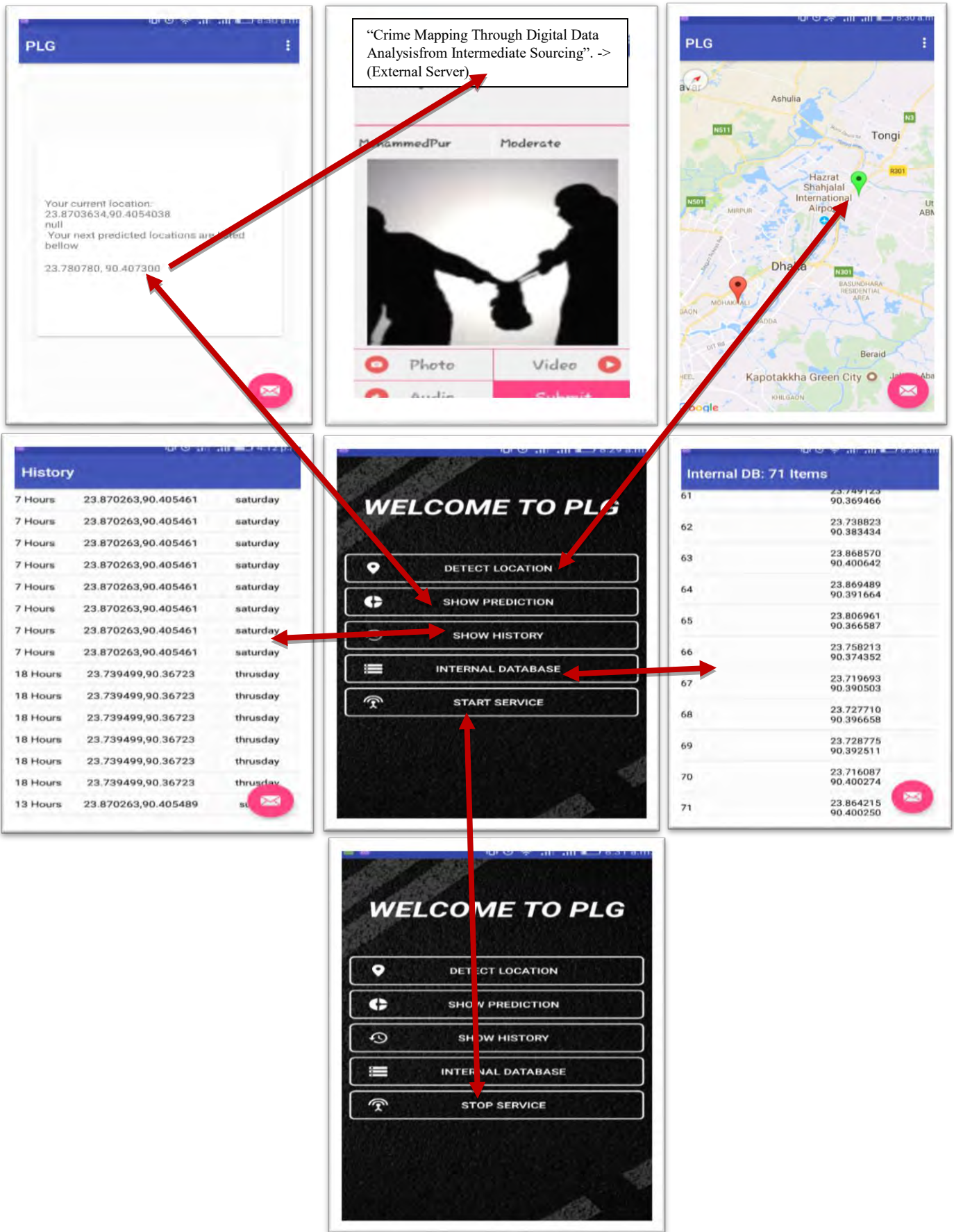


Figure 4.9: Developed Mobile Application

CHAPTER 5

EXPERIMENTAL RESULT ANALYSIS

This chapter shows the outcome for the training and testing datasets. The datasets are used to train and test PLG. The same datasets are also used for training and testing Logistic Regression, Decision Tree and k-NN algorithms for comparing the prediction accuracies generated by the following algorithms.

5.1 PLG

We have got 79.8341% accuracy for the training dataset. We have also used three testing datasets for computing the accuracy level of prediction given by the PLG.

Table 3: Accuracy for the datasets

Dataset	Type	Accuracy
Dataset 0	Training	79.8341%
Dataset 1	Testing	76.5231%
Dataset 2	Testing	77.4667%
Dataset 3	Testing	77.2359%
Average (Testing Datasets)	Testing	77.4085%

As we can see that, we have got the accuracy 76.5231%, 77.4667% and 77.2359% for the dataset 1, dataset 2 and dataset 3 respectively. PLG gives us an average accuracy 77.4085% for the three testing datasets.

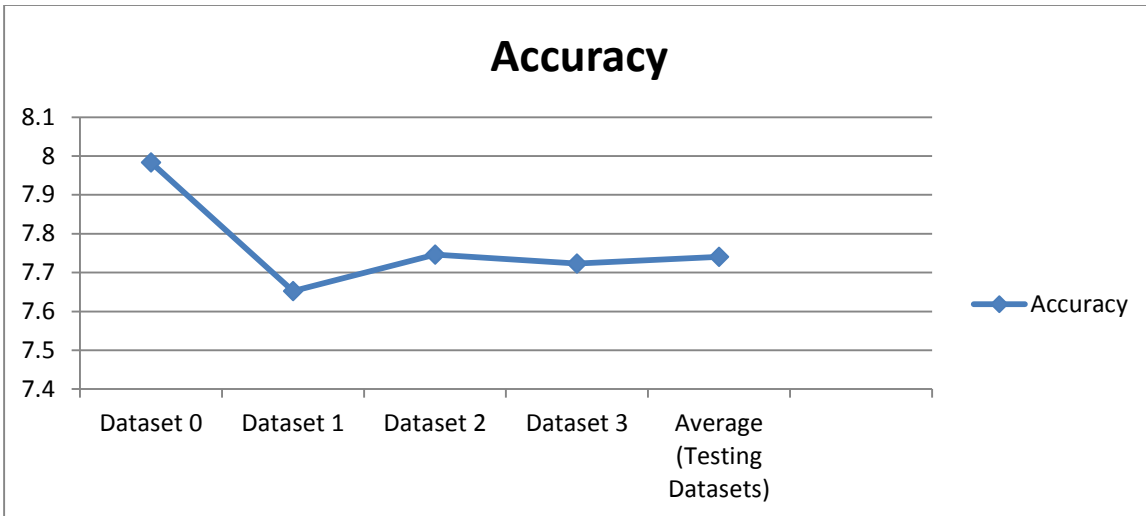


Figure 5.1: Accuracy of PLG on Training & Testing Datasets (Graph)

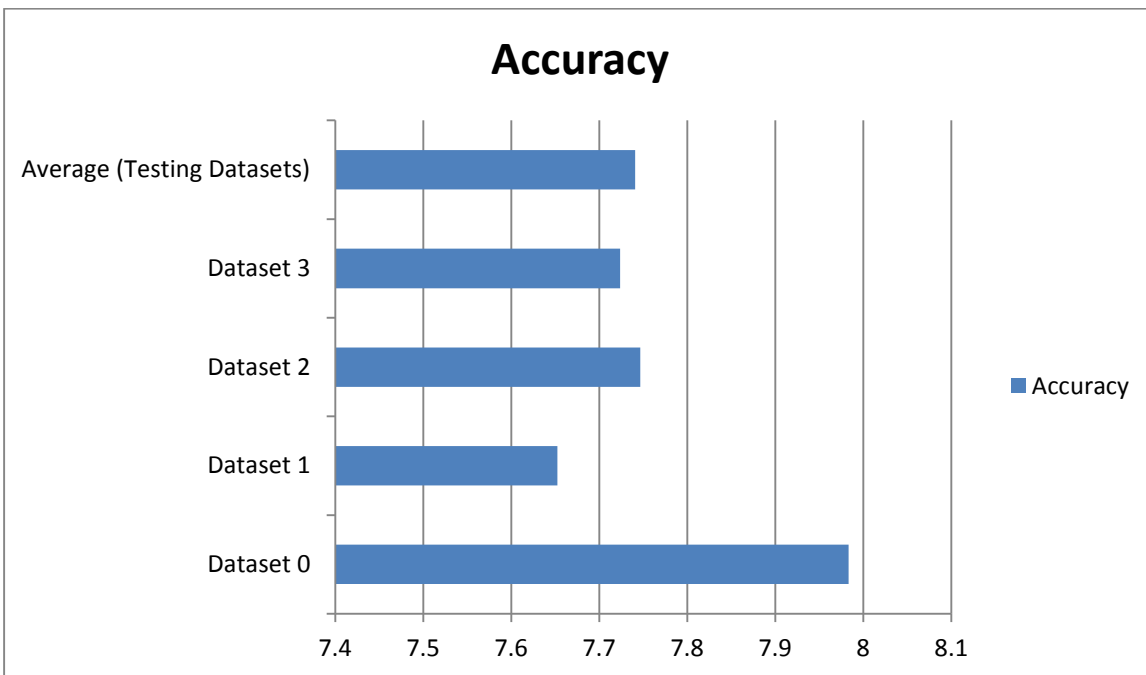


Figure 5.2: Accuracy of PLG on Training & Testing Datasets (Comparison Chart)

From the figures 5.1 and 5.2, we can see the comparisons between the accuracies for the training and testing datasets.

5.2.1 Accuracy of the Testing Datasets:

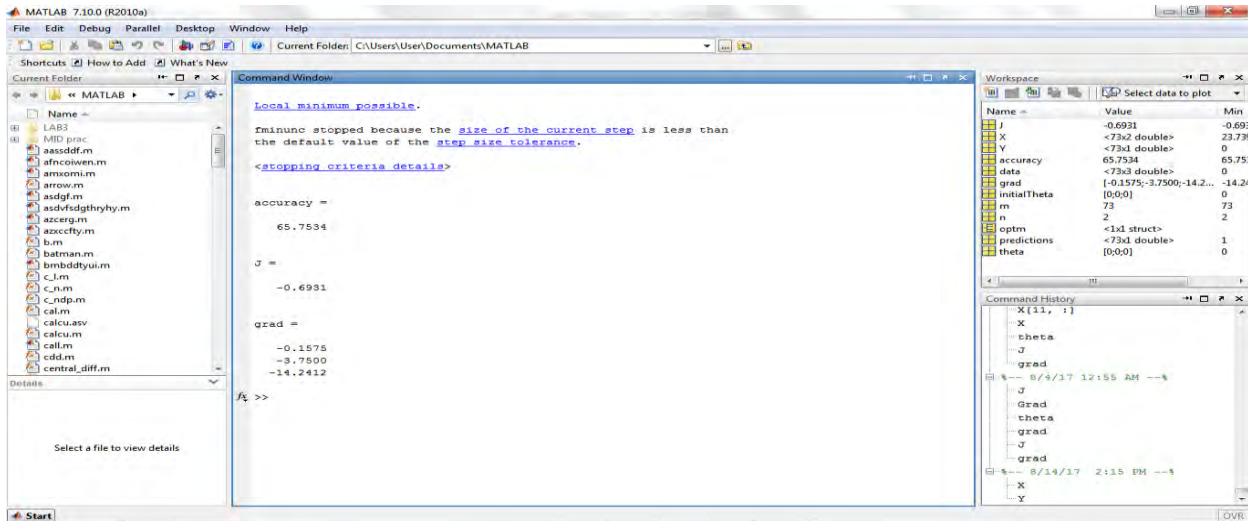


Figure 5.5: Accuracy of the 1st testing dataset (65.7534%)

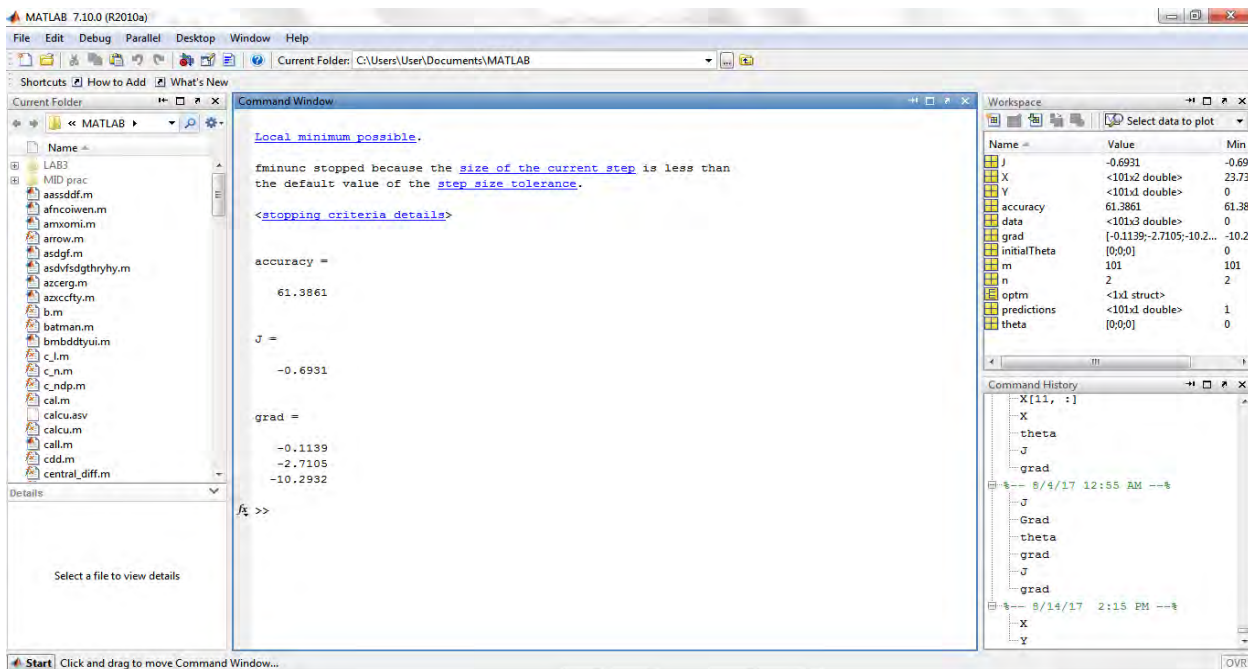


Figure 5.6: Accuracy of the 2nd testing dataset (61.3861%)

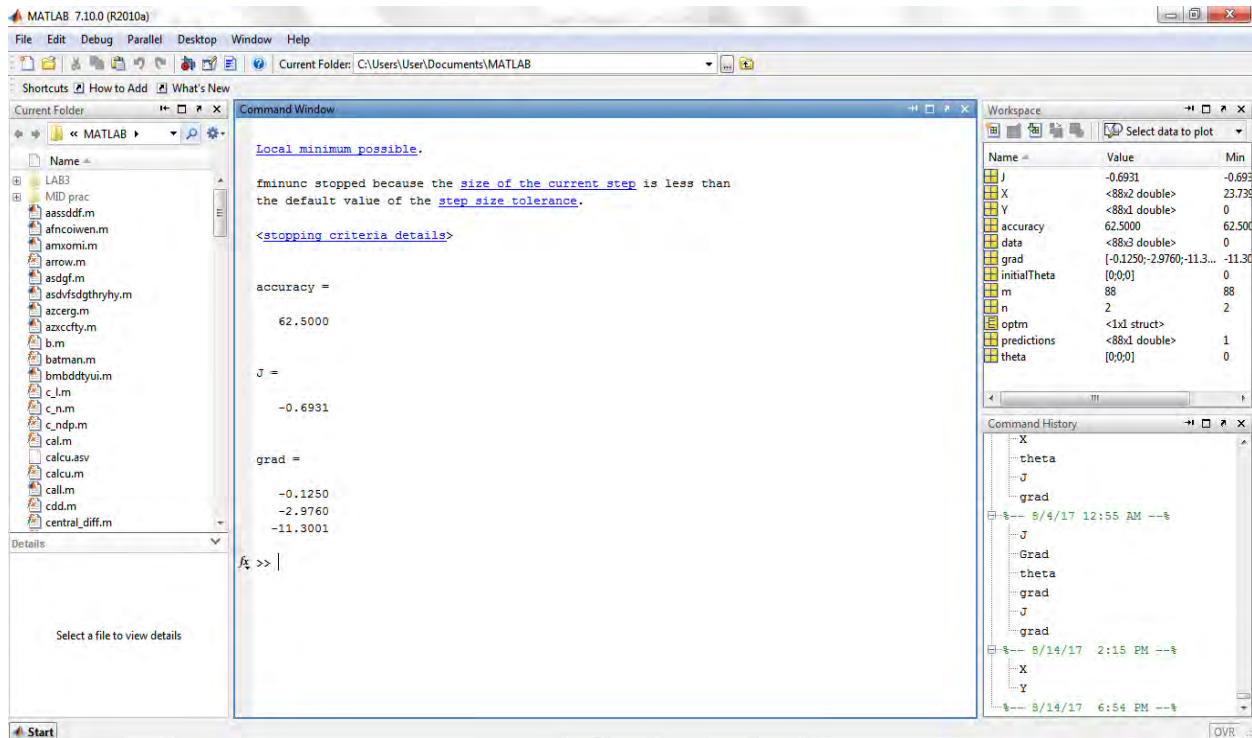


Figure 5.7: Accuracy of the 3rd training dataset (62.5000%)

Table 4: Accuracy for the datasets

Dataset	Type	Accuracy
Dataset 0	Training	71.3740%
Dataset 1	Testing	65.7534%
Dataset 2	Testing	61.3861%
Dataset 3	Testing	62.5000%
Average (Testing Datasets)	Testing	63.213%

The table clearly shows us the accuracies for the training and testing datasets given by the Logistic regression algorithm. The accuracies for the testing datasets are less than the accuracy for the training dataset, which are below 70%.

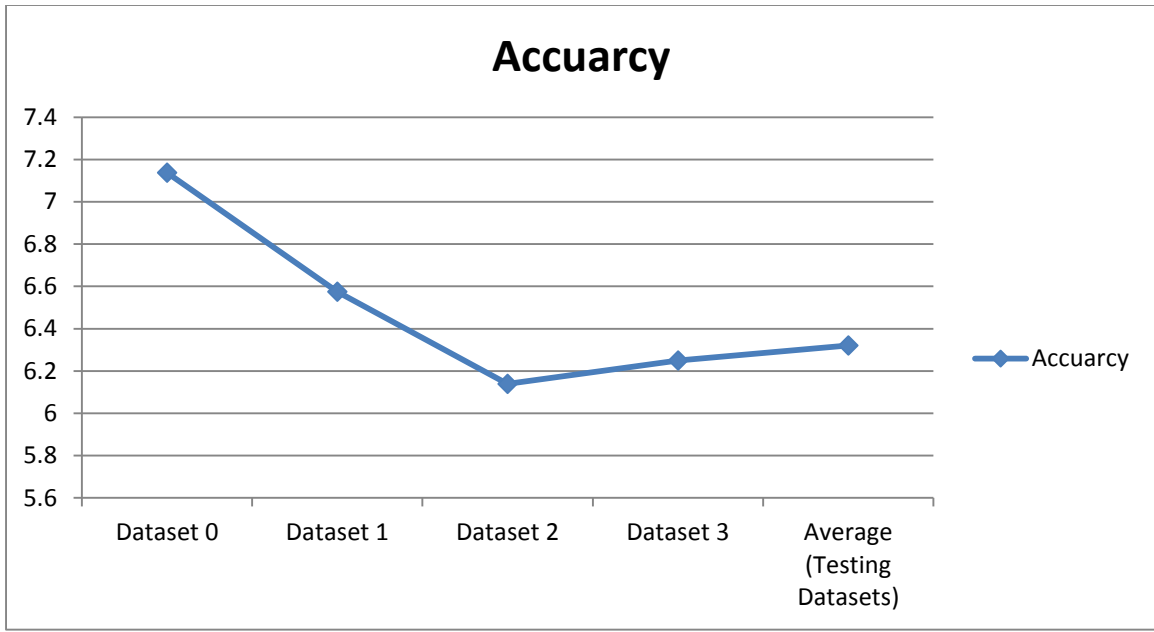


Figure 5.8: Accuracy of Logistic Regression on Training & Testing Datasets (Graph)

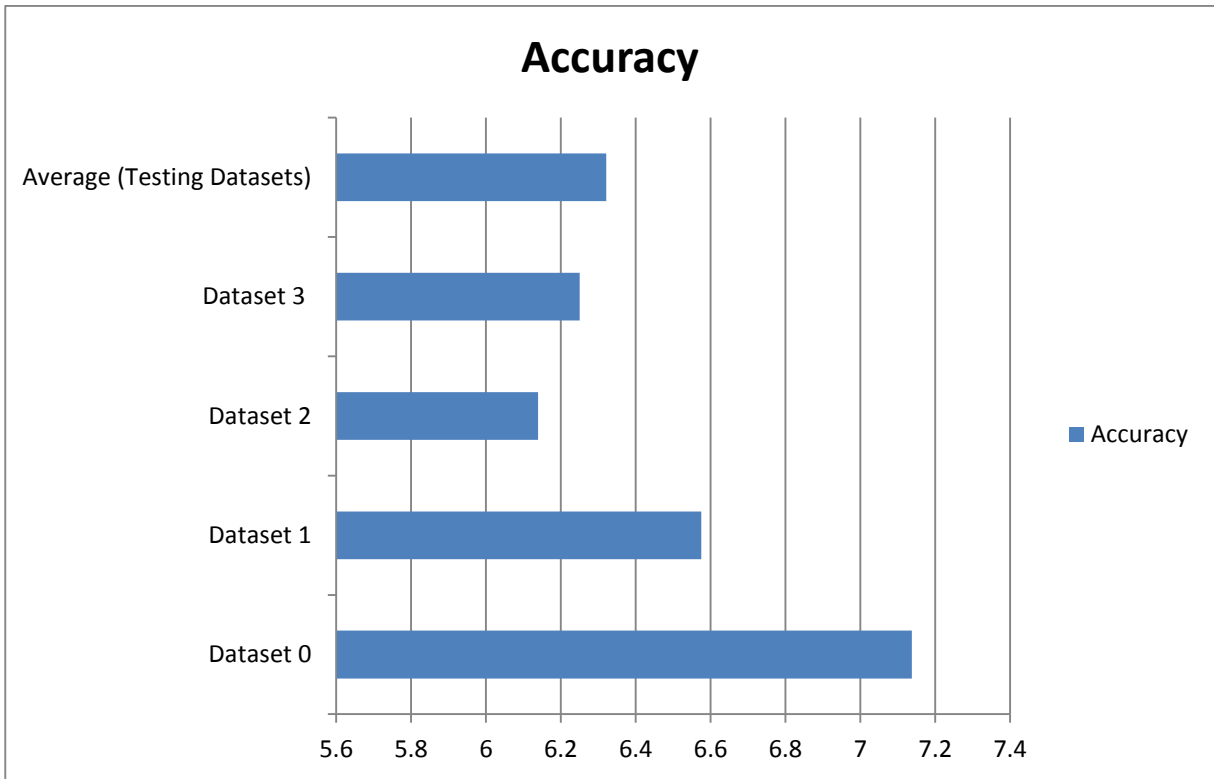


Figure 5.9: Accuracy of Logistic Regression on Training & Testing Datasets (Comparison Bar Chart)

From the figures 5.8 and 5.9, we can see the comparisons between the accuracies for the training and testing datasets, which also show us the decreasing accuracies for the testing datasets.

5.2.2 Comparison with PLG

Table 5: Comparison with PLG (Accuracy %)

Datasets	Logistic Regression	PLG
Dataset 0	71.3740%	79.8341%
Dataset 1	65.7534%	76.5231%
Dataset 2	61.3861%	77.4667%
Dataset 3	62.5000%	77.2359%
Average (Testing Datasets)	63.213%	77.4085%

Table and figures 5.10 and 5.11 show us the accuracies for the same training and testing datasets that have used in Logistic Regression and PLG algorithm. For both kinds of datasets, PLG gives the better accuracy for predicting next possible location than Logistic regression. The average accuracy for PLG is 77.4085%, where the average accuracy for Logistic regression is 63.213%. PLG works more efficiently than Logistic regression here.

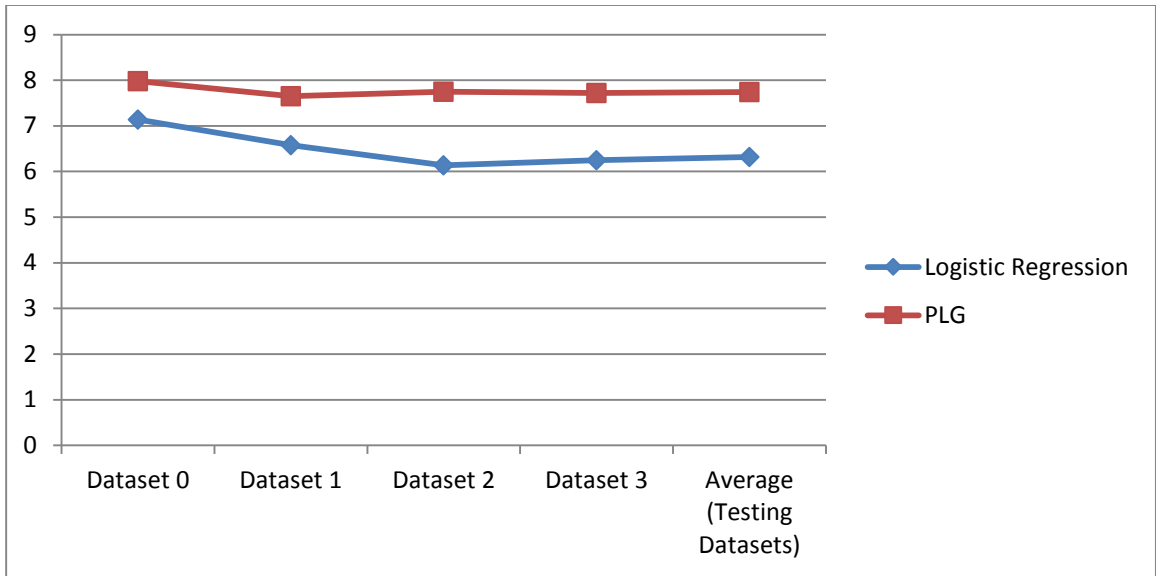


Figure 5.10: Comparison between Logistic Regression & PLG (Accuracy %)

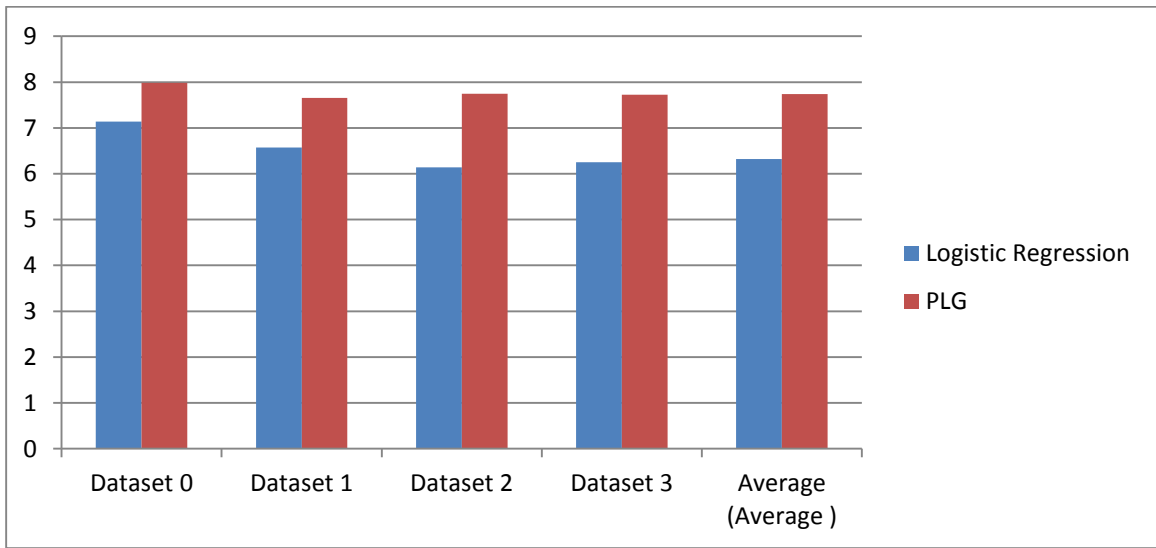


Figure 5.11: Comparison between Logistic Regression & PLG (Accuracy %)

5.3 Decision Tree

We have got accuracy 76.2987% for the training dataset. This is the same training dataset that we have used for training our proposed algorithm (PLG). We have implemented the following Decision tree algorithm on 3 testing datasets to see the accuracy level. These 3 testing datasets are the same datasets that we have used for testing the accuracy of our proposed algorithm (PLG).

Table 6: Accuracy for the datasets

Datasets	Type	Accuracy
Dataset 0	Training	76.2987%
Dataset 1	Testing	73.8259%
Dataset 2	Testing	70.6852%
Dataset 3	Testing	71.5642%
Average (Testing Datasets)	Testing	71.9251%

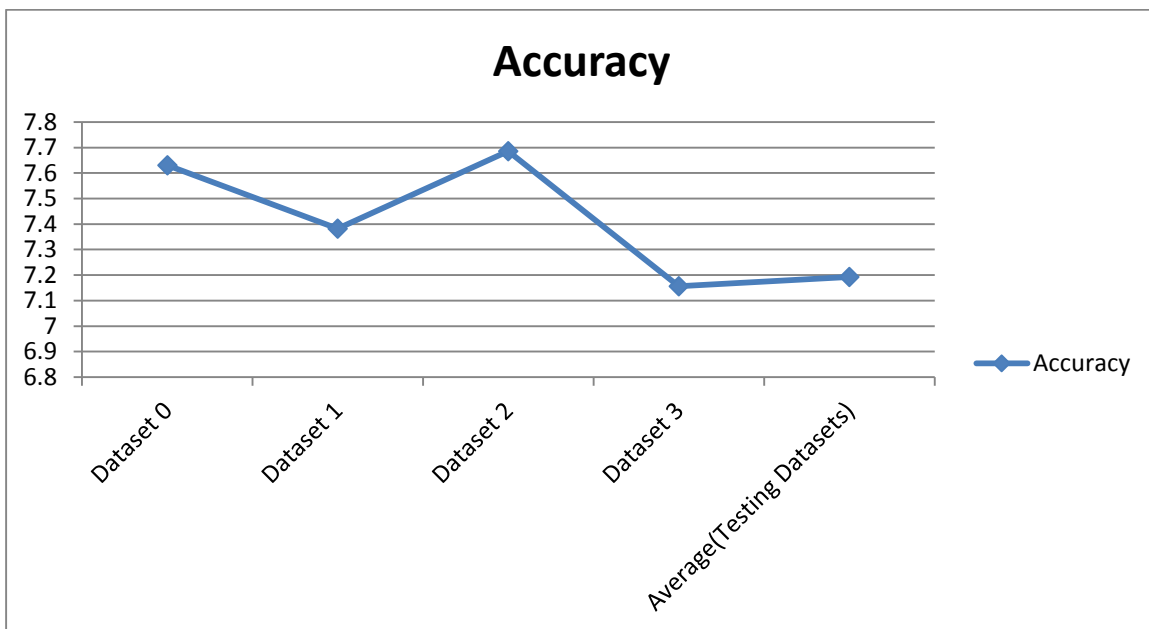


Figure 5.12: Accuracy of Decision Tree on Training & Testing Datasets (Graph)

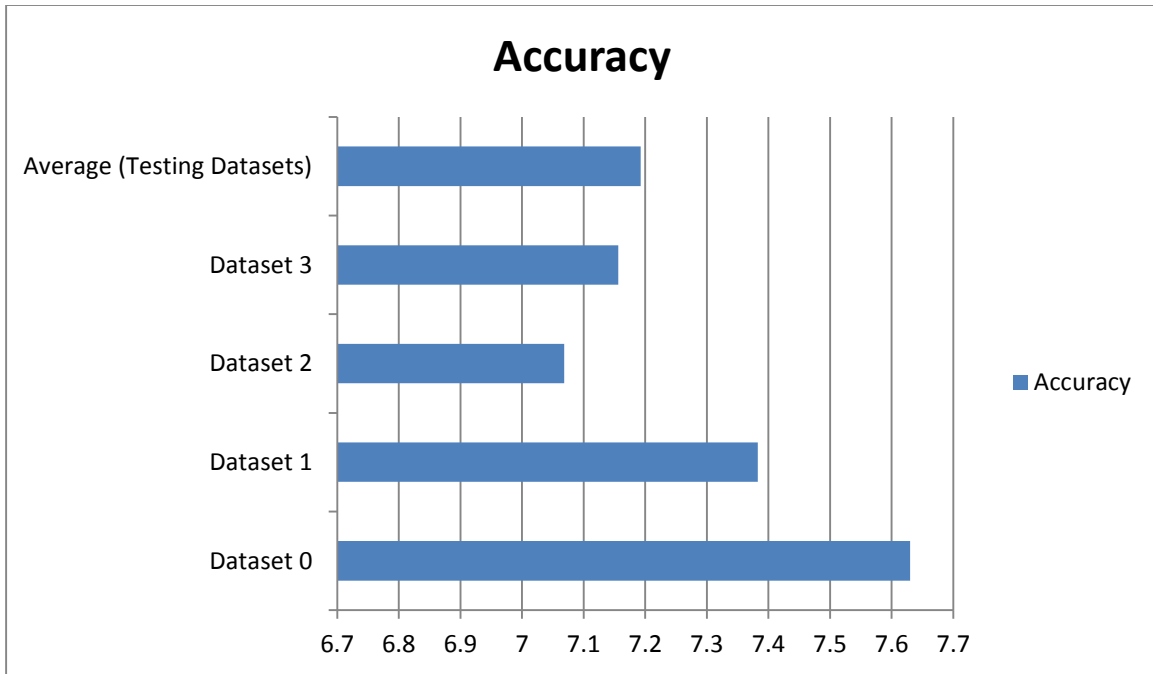


Figure 5.13: Accuracy of Decision Tree on Training & Testing Datasets (Comparison Bar Chart)

The table and comparison figures 5.12 and 5.13 clearly show us the accuracies for the training and testing datasets given by the Decision Tree algorithm. The accuracies for the testing datasets are less than the accuracy for the training dataset, which are below 75%. The average accuracy (testing datasets) of predicting next possible location is 71.9251%.

5.3.1 Comparison with PLG

Table 7: Comparison with PLG (Accuracy %)

Dataset	Decision Tree	PLG
Dataset 0	76.2987%	79.8341%
Dataset 1	73.8259%	76.5231%
Dataset 2	70.6852%	77.4667%
Dataset 3	71.5642%	77.2359%
Average (Testing Datasets)	71.9251%	77.4085%



Figure 5.14: Comparison between Decision Tree & PLG (Accuracy %)

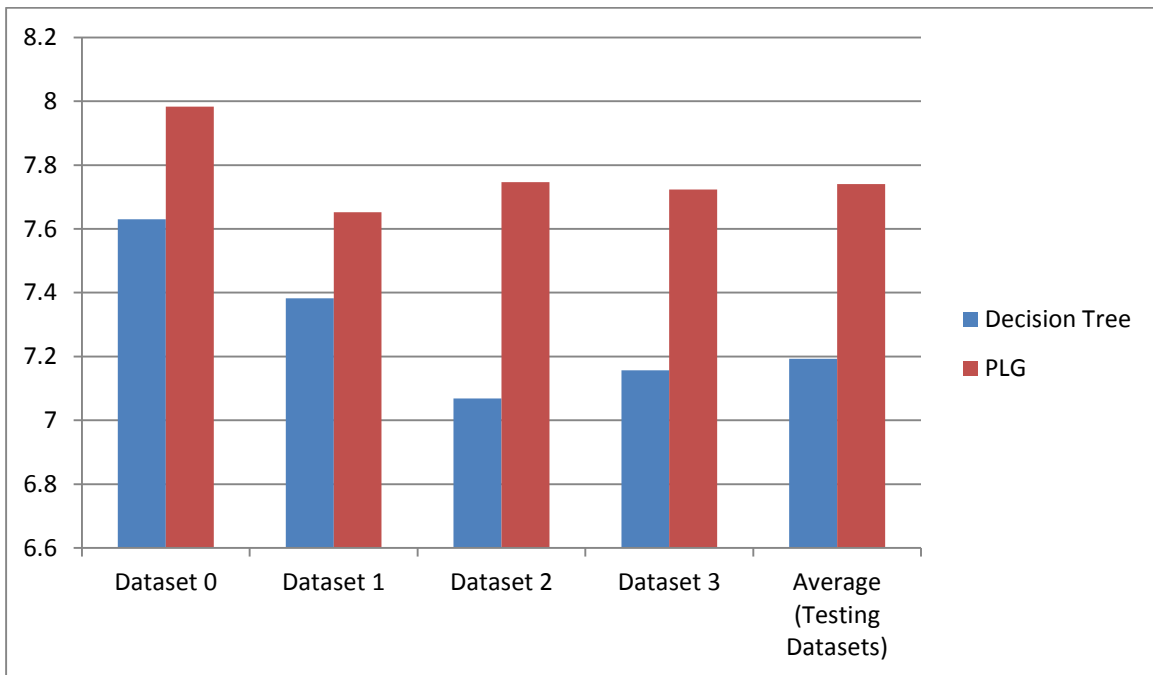


Figure 5.15: Comparison between Decision Tree & PLG (Accuracy %)

Table and figures 5.14 and 5.15 shows us the accuracies for the same training and testing datasets that have used in Decision Tree and PLG algorithm. For both kinds of datasets, PLG gives the better accuracy for predicting next possible location than Decision Tree algorithm. The average accuracy for PLG is 77.4085%, where the average accuracy for Decision Tree is 71.9251%. PLG works more efficiently than Decision Tree here.

5.4 k-NN

We have implemented k-NN (K-Nearest Neighbor) algorithm in order to determine the accuracy by trial and error method. The method represents by continuous, repeated and varied attempts which repeats until finding the best result. Here, the best result is the accuracy in terms of number of iterations. There are some comparative results:

Our dataset is based on the GPS co-ordinates. We have 1 training dataset and 3 testing datasets. At first, we try to create a classifier with the training dataset and gather information of 33 iterations that means $k=1,2,3,4,\dots,33$ (the number of neighbors).

5.4.1 k-NN Parameters:

- 1. $k = 1, 2, 3, \dots, 33$*
- 2. $Cross\ validate = True$*
- 3. $Debug = True$*
- 4. $Distance\ weighting = weight\ by\ 1\ distance$*
- 5. $Mean\ squared = True$*
- 6. $No\ normalization = False$*

With the step size of 1.0, the training accuracy reached the saturation at around 0.64, which is not really over fitting

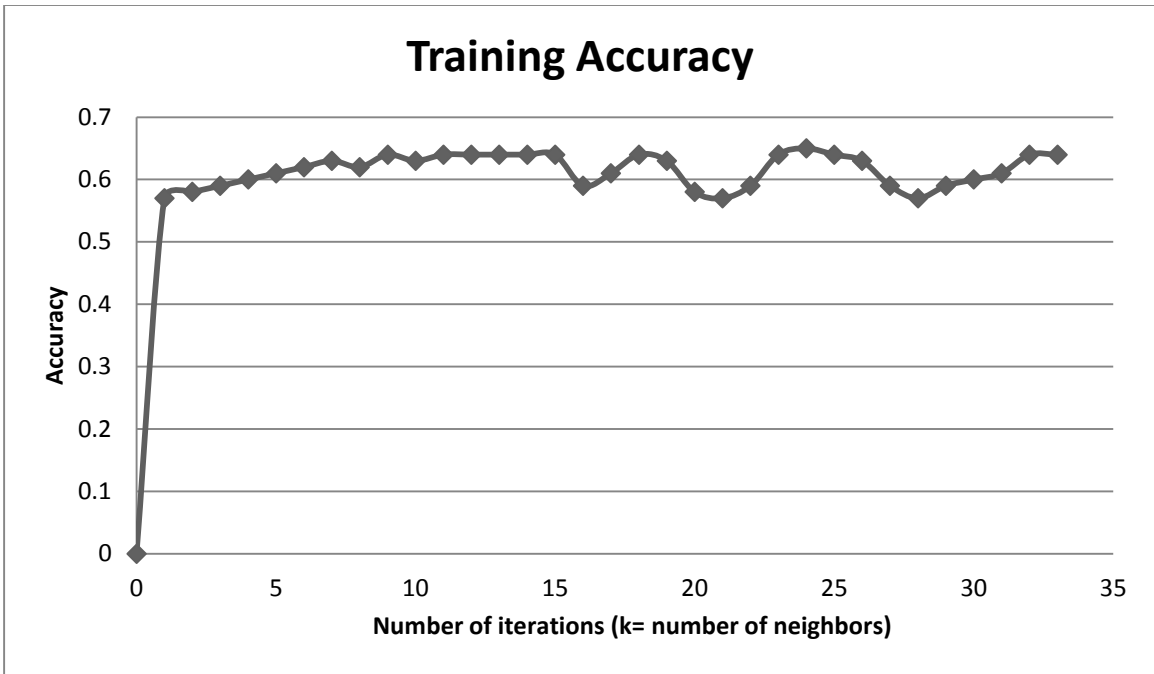


Figure 5.16: Training Accuracy

Before the training took place, we take 10% of the training data as a validation set which will be compared to testing dataset and from that we will find prediction and the k-iteration count for creating the classifiers.

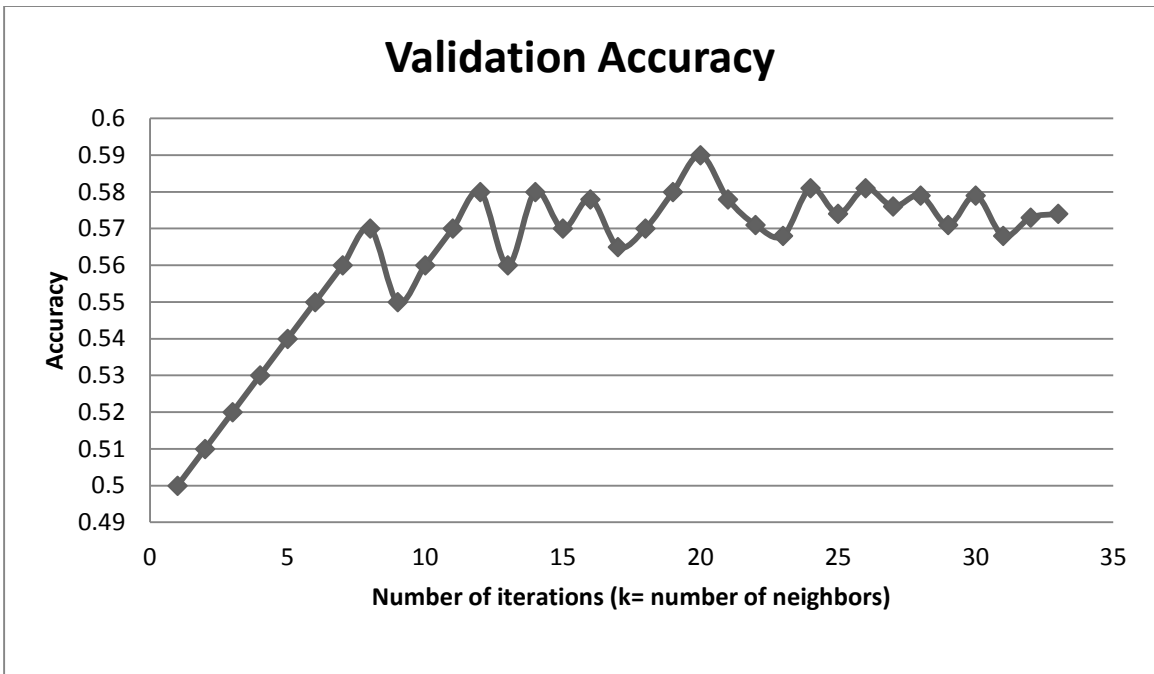


Figure 5.17: Validation Accuracy

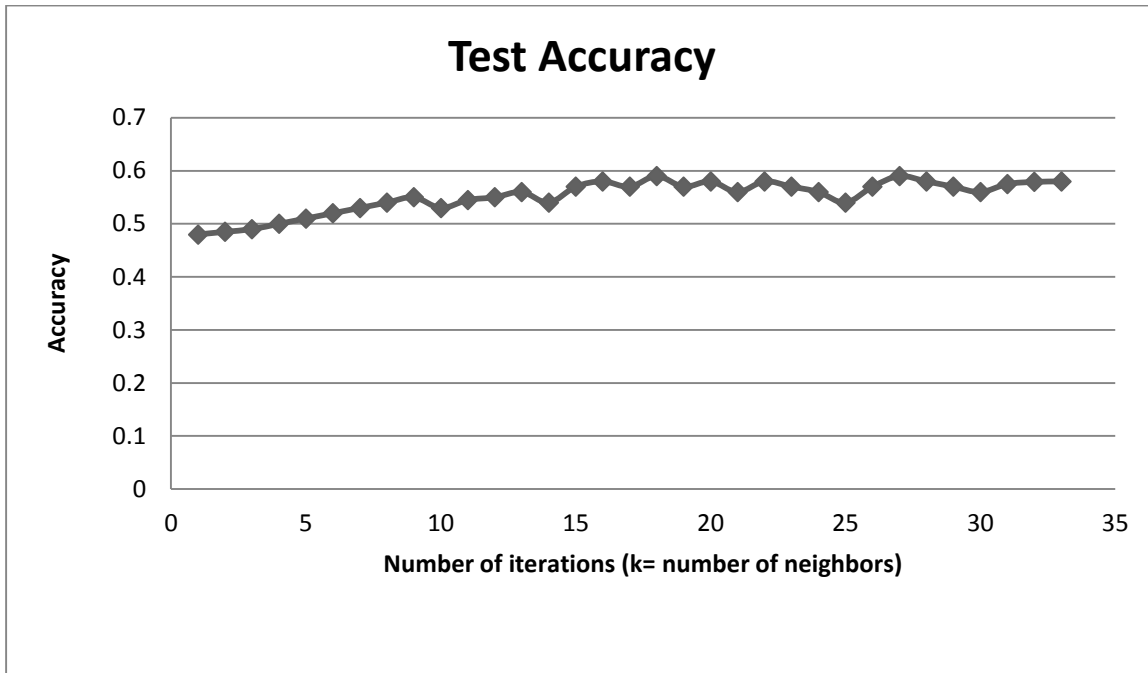


Figure 5.18: Test Accuracy

We have also evaluated accuracy with table so that we can easily compare with other predictive model for accuracy comparison.

Table 8: Accuracy for training and testing datasets

Dataset	Type	Accuracy (%)	Average Accuracy for Testing Dataset (%)
Dataset 0	Training	64.28	60.89
Dataset 1	Testing	57.28	
Dataset 2	Testing	65.87	
Dataset 3	Testing	59.53	

We have also calculated the accuracy for bar diagram that shows the comparative analysis of different datasets.

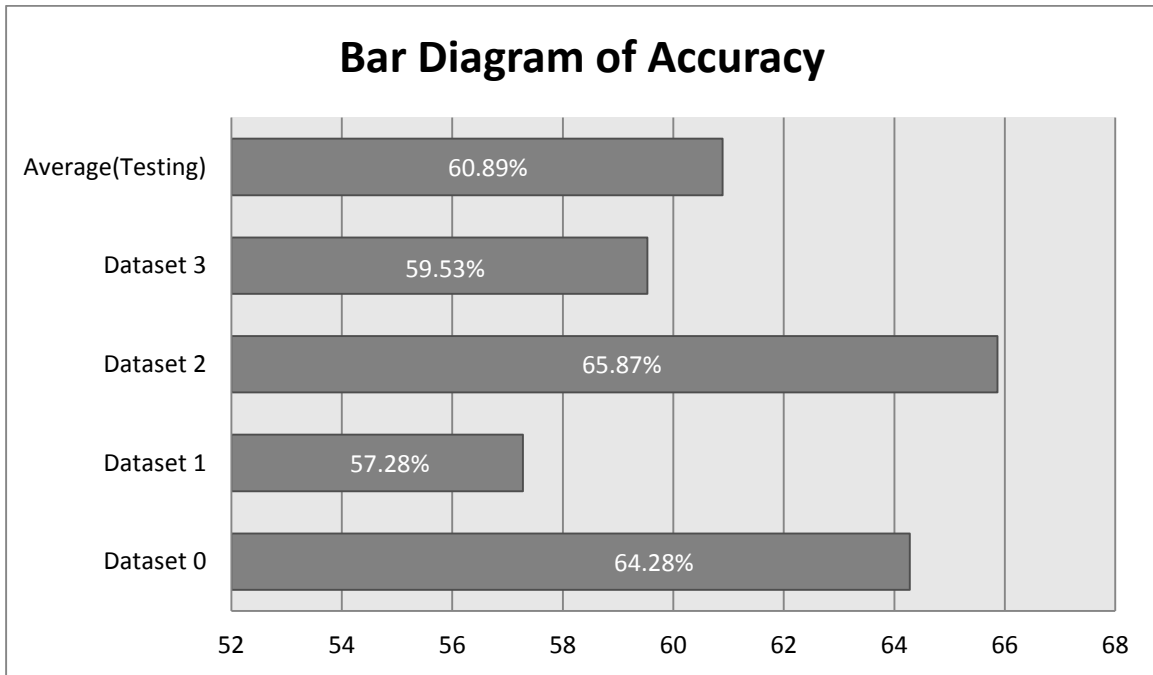


Figure 5.19: Bar Diagram of accuracy for training and testing datasets

We have also calculated the accuracy for line diagram that shows the comparative analysis of different datasets.

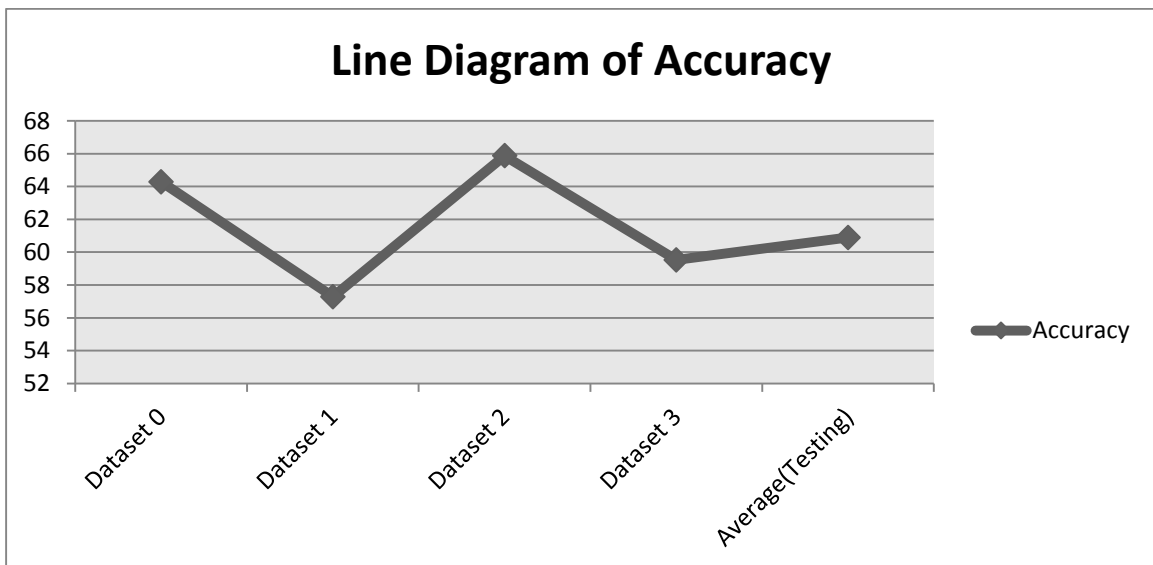


Figure 5.20: Line Diagram of accuracy for training and testing datasets

5.4.2 Comparison with PLG

Table 9: Comparison with PLG (Accuracy %)

Datasets	k-NN	PLG
Dataset 0	64.28%	79.8341%
Dataset 1	57.28%	76.5231%
Dataset 2	65.87%	77.4667%
Dataset 3	59.53%	77.2359%
Average (Testing Datasets)	60.89%	77.4085%

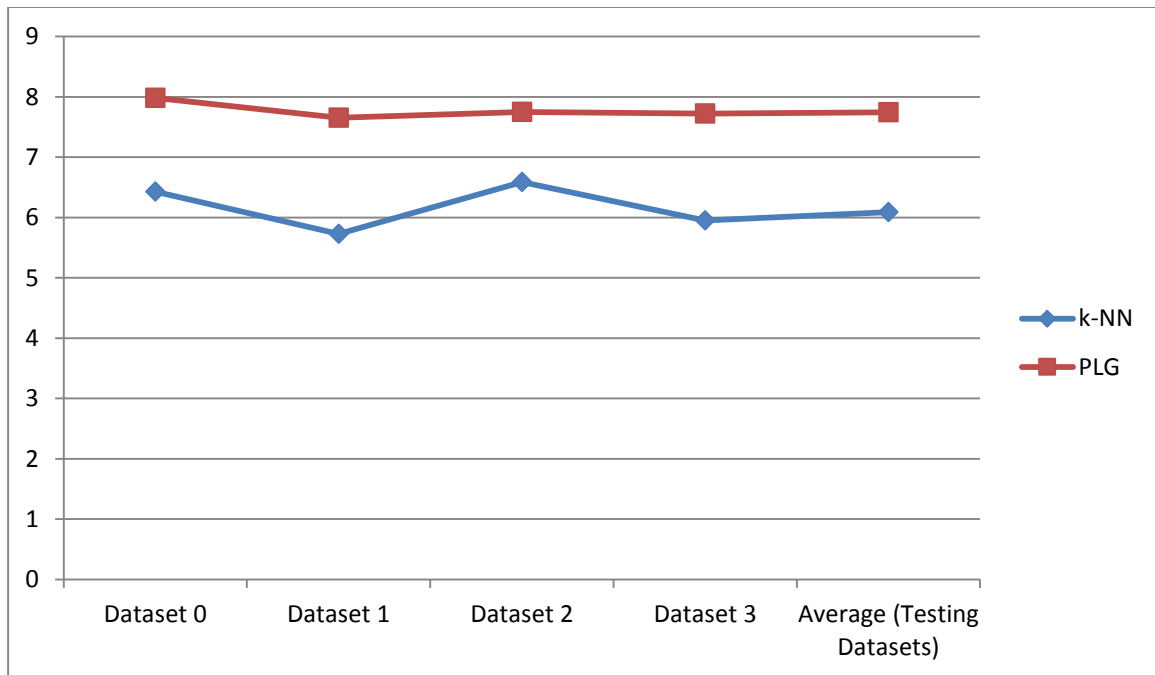


Figure 5.21: Comparison between k-NN & PLG (Accuracy %)

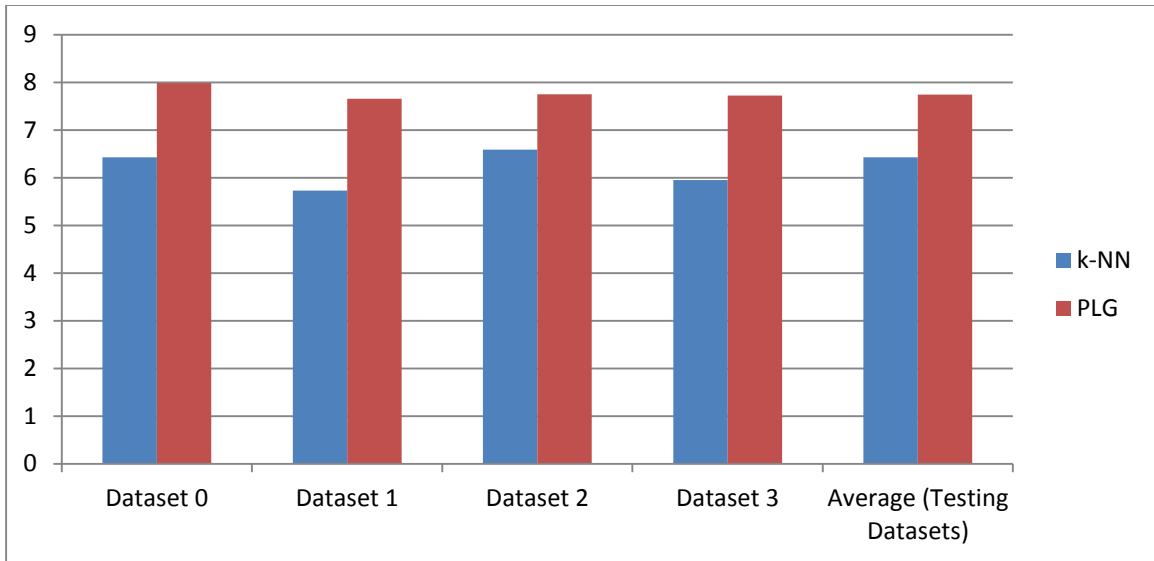


Figure 5.22: Comparison between k-NN & PLG (Accuracy %)

Table and figures 5.21 and 5.22 shows us the accuracies for the same training and testing datasets that have used in k-NN and PLG algorithm. The average accuracy for PLG is 77.4085%, where the average accuracy for k-NN algorithm is 60.89%. PLG works more efficiently than k-NN here.

5.5 Comparisons between the algorithms

Table 10: Comparison based on accuracy

Datasets	PLG	Logistic Regression	Decision Tree	k-NN
Dataset 0	79.8341%	71.3740%	76.2987%	64.28%
Dataset 1	76.5231%	65.7534%	73.8259%	57.28%
Dataset 2	77.4667%	61.3861%	70.6852%	65.87%
Dataset 3	77.2359%	62.5000%	71.5642%	59.53%
Average (Testing)	77.4085%	63.213%	71.9251%	60.89%

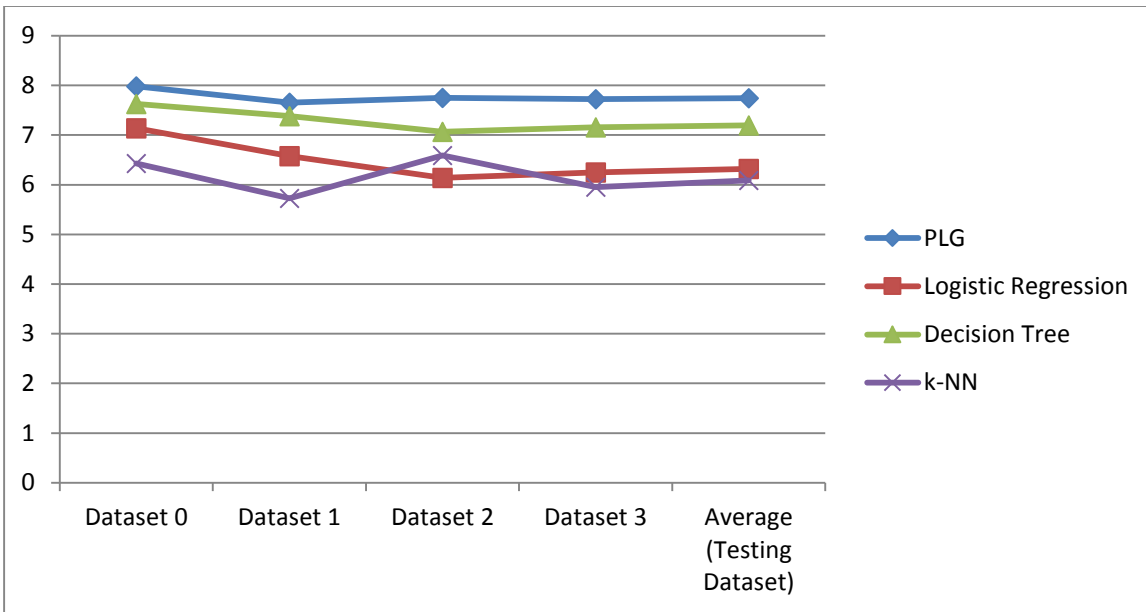


Figure 5.23: Comparisons between algorithms (accuracy)

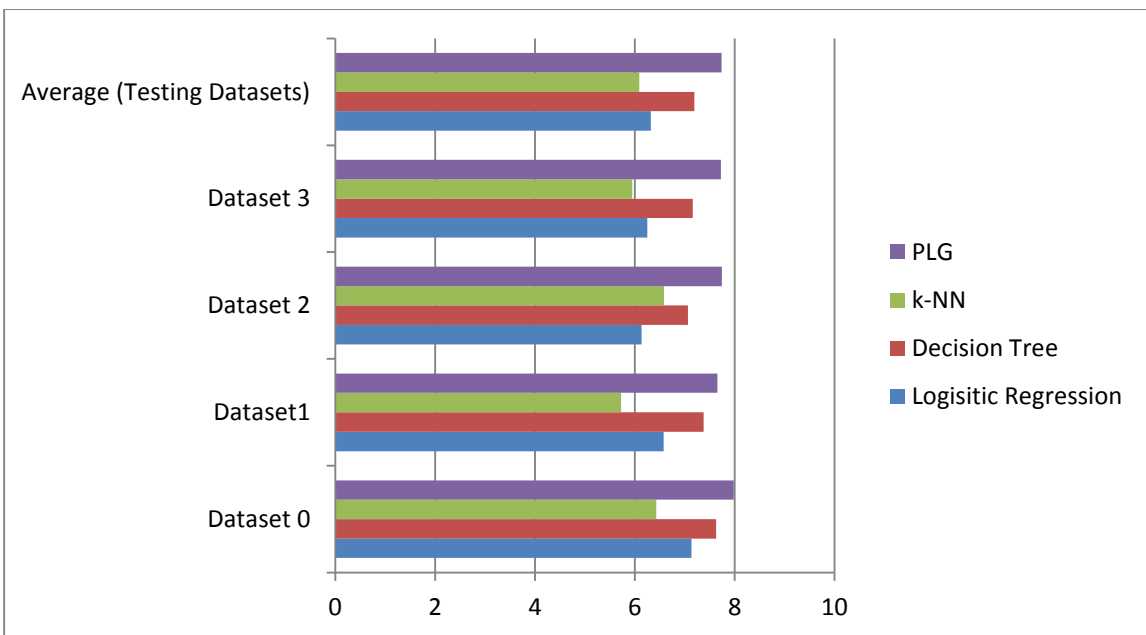


Figure 5.24: Comparisons between algorithms (accuracy)

From the table and figures 5.23 and 5.24, we can see the comparisons based on accuracies between PLG, k-NN, Decision Tree and Logistic Regression. For the training dataset (Dataset

0), PLG provides the accuracy for prediction is 79.8341%, Logistic regression provides 71.3740%, Decision Tree provides 76.2987% and k-NN provides 64.28%. So, it is clearly seen that, PLG has provided better accuracy while it's training period.

Furthermore, during the testing period, PLG along with Logistic regression, Decision Tree and k-NN have been tested on the datasets Dataset 1, Dataset 2, Dataset 3. In the testing session, we have found different accuracies for the following algorithms. For dataset 1, PLG provides 76.5231%, Logistic regression provides 65.7534%, Decision tree provides 73.8259% and k-NN provides 57.28% accuracy for predicting next possible location. Next, for Dataset 2, PLG provides 77.4667% accuracy for predicting next possible location, which indicates the increasing efficiency of accuracy for predicting next possible location. On the other hand, accuracies for Logistic regression, Decision tree and k-NN are 61.3861%, 70.6852% and 65.87% respectively. So, PLG is working well for dataset 2 also. Moreover, for Dataset 3, PLG provides an accuracy of 77.2359% for predicting next possible location, which indicates the efficiency level of the PLG in terms of predicting locations. Besides, accuracies for Logistic regression, Decision tree and k-NN are 62.5000%, 71.5642% and 59.53% respectively. So, PLG is giving better prediction on Dataset 3 comparing to other three algorithms. Now, we have calculated average accuracy for testing Datasets (1, 2, and 3) for the four algorithms. We have got average accuracy 77.4085% for PLG in terms of predicting next possible location. On the other hand, we have got average accuracy 63.213%, 71.9251% and 60.89% for Logistic regression, Decision tree and k-NN respectively in terms of predicting next possible location. So, from the average accuracies, we can clearly see that, PLG is predicting efficiently and better than Logistic regression, Decision tree and k-NN algorithms.

CHAPTER 6

CONCLUSION

This chapter contains the Conclusion Remarks and Future Works, which will give the summary of our thesis work and also give the indication of our future plan with our thesis project.

6.1 Conclusion Remarks

In conclusion, we can say that our system can predict the future location of the system users by using Machine Learning. The prediction is pretty much accurate and its accuracy is near 79%, which is better comparing to other machine learning algorithms such as Decision tree, KNN, Logistic Regression for our training and testing datasets. By using our system, user can easily know about the traffic update to his next destination. He can also know about the restaurants, shopping malls, crime updates of his next predictive destination. Our system also secures the identity of the users from the 3rd party intervention. In the summation, we hope that, our system will be very helpful for the user in terms of saving time and getting useful information about the next predictive location of the user.

6.2 Future Works

1. We will improve our algorithm for behavioral prediction, which will give more accurate results and predictions.
2. We will improve our server side works so that data will be retrieved from the server more efficiently.
3. We will improve and increase the security of the application and server.
4. We will work on data prediction accuracy and will try to increase the accuracy more.

REFERENCE

1. Huang W, Li M, Hu W, Song G, Xie K, “Hierarchical Destination Prediction Based on GPS History”, in *10thInternational Conference on Fuzzy Systems and Knowledge Discovery*, 2013. *FSKD'13. IEEE*. IEEE, 2013, pp. 972-977.
2. Binh T. NguyennNhan V. Nguyen, Nam T. Nguyen, Nam T. Nguyen, Huyen T. Tran, “A Potential Approach for Mobility Prediction Using GPS Data”, in *Seventh International Conference On Information Science and Technology, Vietnam, Aprill 16-19, 2017. SICIST'17. IEEE*. IEEE, 2017, pp. 45-50.
3. Ashbrook D, Starner T, “Learning Significant Locations and Predicting User Movement with GPS”, in *International Symposium on Wearable Computers, 2002. ISWC'02. IEEE*. IEEE, 2002, pp. 101-108.
4. Cao L, She J, “Can your Friends Predict Where You will be?”, *Proceedings of the IEEE International Conference on Internet of Things (iThings)*, pp. 450-455, 2014.
5. Academic Solutions, Directory of Statistical Analyses, Regression Analysis, Statistics Solutions, <http://www.statisticssolutions.com>. Retrieved on August 1, 2017.
6. Logistic Regression, UFLDL Tutorial. <http://ufldl.stanford.edu>. Retrieved on August 6, 2017.
7. Logistic Regression implementation. YOUTUBE. <https://www.youtube.com>. Retrived on August 6, 2017.
8. Niu X, Zhu Y, Zhang X, “Deepsense: A Novel Learning Mechanism for Traffic Prediction with Taxi GPS Traces”, in *GLOBECOM 2014 – Symposium on Selected Areas in Communications: GC14 SAC Internet of Things, 2014. GC14: SAC'14. IEEE*. IEEE, 2014, pp. 2745-2750.
9. Community, MathWroks®. <https://www.mathworks.com>. Retrieved on August 4, 2017.
10. Stackoverflow. <https://stackoverflow.com>. Retrieved on August 4, 2017.
11. Stack Exchange, Cross Validated. <https://stats.stackexchange.com>. Retrieved on August 5, 2017.
12. Algorithms and Ideas in Java. <https://intelligentjava.wordpress.com>. Retrieved on July 29, 2017.
13. Machine Learning I: Decision trees UMBC CSEE. <https://www.csee.umbc.edu>. Retrieved on July 26, 2017.

14. Jabbar M. A, Deekshatulu B.L, Chandra P, "Classification of Heart Diseases Using K-Nearest Neighbor and Genetic Algorithm", in *International Conference on Computational Intelligence: Modeling Techniques and Applications, 2013. CIMTA'13*. pp. 86-94.
15. Zhang Z, "Introduction to machine learning: k-nearest neighbor", Big-data Clinical Trial Column. <https://www.ncbi.nlm.nih.gov>. Retrieved on August 1, 2017.
16. KNN classification, ResearchGate. <https://www.researchgate.net>. Retrieved on August 2, 2017.
17. Trial And Error, Wikipedia. <https://en.wikipedia.org>. Retrieved on August 3, 2017.
18. k-nearest neighbor algorithm, Wikipedia. <https://en.wikipedia.org>. Retrieved on August 3, 2017.
19. 10: Advice for applying Machine Learning, HoleHouse. <http://www.holehouse.org>. Retrieved on August 10, 2017.
20. A Beginner's Guide to Neural Networks: Part Two, Medium. <https://medium.com>. Retrieved on June 5, 2017.
21. Role of bias in neural networks, StackOverflow. <https://stackoverflow.com>. Retrieved on August 9, 2017.
22. Deep Learning Feedforward Neural Network, Medium. <https://medium.com>. Retrieved on May 9, 2017.
23. A List of Cost Functions Used in Neural Networks, Alongside Applications, Medium. <https://stats.stackexchange.com>. Retrieved on August 5, 2017.
24. Neural Networks, Webpages. <http://www.webpages.ttu.edu>. Retrieved on July 12, 2017.
25. Chap2, NeuralNetworksAndDeepLearning. <http://neuralnetworksanddeeplearning.com>. Retrieved on August 1, 2017.
26. Supervised and unsupervised machine learning algorithms, MachineLearningMastery. <http://machinelearningmastery.com>. Retrieved on June 13, 2017.
27. A Basic Introduction to Neural Networks, PagesCsWisc. <http://pages.cs.wisc.edu>. Retrieved on July 15, 2017.
28. Aunamika A. I, Tithi S. S, Dev P, "Crime Mapping Through Digital Data Analysis From Intermediate Repository", BRACU, 2016, pp. 9-46.