

**Hyper-Optimized Machine Learning and Deep Learning Methods for
Geo-Spatial and Temporal Function Estimation**

by

NEELABH PANT

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August, 2018

Copyright © by Neelabh Pant 2018

All Rights Reserved

To my father, mother, sister and Sai

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Ramez Elmasri for constantly motivating and encouraging me, and also for his invaluable advice during the course of my doctoral studies. I wish to thank my committee members Dr. Leonidas Fegaras, Dr. Sharma Chakravarthy, Mr. David Levine and Dr. Shashi Shekhar for their interest in my research and for taking time to serve in my dissertation committee.

I am grateful to all the professors who taught me during my graduate studies at The University of Texas at Arlington. Finally, I would like to express my deep gratitude to my father, mother, sister and Sai for their sacrifice, encouragement, inspiration and endless love during my PhD work. I am also extremely grateful to several of my friends who have helped me throughout my career.

May 29, 2018

ABSTRACT

Hyper-Optimized Machine Learning and Deep Learning Methods for Geo-Spatial
and Temporal Function Estimation

Neelabh Pant

The University of Texas at Arlington, 2018

Supervising Professor: Ramez Elmasri

Owing to a high degree of freedom in human mobility, accurate modelling/estimation of human mobility function remains a challenge. Numerous work in the literature have tried to address the challenge using various traditional machine learning methods on spatio-temporal attributes of data. We compare the use of Varied-K Means clustering, Hidden Markov Model techniques, feed forward neural networks, recurrent neural networks (RNN) and Long Short Term Recurrent Neural Networks (LSTM) to predict a user’s future movement based on the user’s past historical data. Although several techniques were proposed to predict a user’s movement, not many have concentrated on a user’s location based on weekday and time period within the day, as well as other features such as weather conditions (for example, temperature and precipitation). We introduce machine learning and deep neural network models using regression and classification techniques that can answer **day-specific queries** like “*Where is the user most likely to be on a specific day of the week*”, **day-time specific queries** like “*Where is the user most likely to be on a specific day of the*

week and time of the day” or **spatio-temporal-weather** queries like “*The user is currently at a specific location with temperature of 70 degree and precipitation < 0.1 , where would the user most likely to travel next given a day of the week and time of the day*”. Our deep learning classifier gives an average classification accuracy of 88%, which is almost 1.4 times better than other traditional machine learning methods. Deep learning regression’s loss constantly keeps on decreasing as we train the model more.

We then shift our domain from geospatial data to financial data by introducing the famous problem of predicting future stock prices and future currency exchange rates. Researchers have done extensive work in creating models to predict stock prices but to the best of our knowledge most of the works have not shown techniques to optimize the hyperparameters and find “the best” model out of all the possible models to predict stock prices as close as possible. In this work we have worked on a meta-heuristic hyperparameter optimization technique called Genetics Algorithm / Evolutionary Algorithm through which we have selected the best model for a specific kind of problem. We also make use of the sliding window technique to capture the patterns within the data for better prediction. We decide the size of window more by calculating the partial-auto-correlation between the data to calculate the best window size. In our temporal analysis we chose to predict future Apple stock prices using technical and fundamental analysis (hybrid approach). By making use of hyperparameter optimization using genetics we try to compare different artificial neural networks among themselves and try to find the best model with the right hyperparameters for a certain kind of problem. Our method also determines the most important features (parameters) for the accurate prediction of future stock prices.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
ABSTRACT	v
LIST OF FIGURES	ix
LIST OF TABLES	xi
Chapter	Page
1. INTRODUCTION	1
1.1 Human Mobility Function Estimation	1
1.2 Stocks and Currency Exchange Prediction	4
1.3 Dissertation Organization	5
2. Detecting Meaningful Places and Predicting Locations Using Varied K- Means and Hidden Markov Model	7
2.1 Introduction	7
2.2 Related Works	9
2.3 Clustering of Locations	10
2.4 Hidden Markov Model	14
2.5 Experiments	20
2.6 Conclusion	23
3. Deep Learning Models to Predict User’s Spatial Locations Using GPS Data	26
3.1 Introduction	26
3.2 Related Works	28
3.3 Methodology	29
3.3.1 Multiple Linear Regression for Predicting Location Using ANN	29

3.3.2	Location classification using ANN	31
3.3.3	Optimization Algorithm and Regularization	34
3.4	Analysis: American Dataset	36
3.4.1	GPS Data Sampling Rate	37
3.4.2	Hour-of-day - Movement Linking	37
3.4.3	Weather and precipitation Correlation with movement	38
3.5	Experiments	41
3.6	Conclusion and Future Work	45
4.	Hyper Optimized ANNs, RNNs and LSTMs with Sliding Window to Predict Short Term Apple's Stocks Prices, Currency Exchange and GPS Locations	49
4.1	Introduction and Related Works	49
4.2	Methodology for constructing and optimizing models	54
4.2.1	Four Main Characteristics of Each Experiments	54
4.2.2	Artificial Neural Network	56
4.2.3	Recurrent Neural Network	57
4.2.4	Long Short-Term Memory Neural Network	59
4.2.5	Hyper-Parameters Optimization	61
4.3	Experimental Results	61
4.3.1	Stocks Prediction	61
4.3.2	Currency Exchange	67
4.3.3	Location Prediction	73
4.4	Conclusion	77
5.	CONCLUSIONS	79
5.1	Summary of Contributions	79
	REFERENCES	81
	BIOGRAPHICAL STATEMENT	88

LIST OF FIGURES

Figure	Page
2.1 Graph to identify meaningful locations.	11
2.2 Clusters found for user3 when radius=0.2 miles	13
2.3 Transition Between Clusters	15
2.4 Transition Between Clusters with Probabiities	15
2.5 Hidden Markov Model for Days In a Week	18
2.6 Hidden Markov Model for Days and Time	19
2.7 Cluster 208	21
2.8 Cluster 195	22
3.1 Feed Forward Neural Network (Regression)	30
3.2 Sigmoid Function	31
3.3 Dummy Variables	33
3.4 Sampling Rate	38
3.5 Time VS Day	39
3.6 Temperature	40
3.7 precipitation	41
3.8 User Locations for 6 Months	42
3.9 Frequent user Locations for 6 Months	43
3.10 Training MSE	44
3.11 Validation MSE	44
3.12 Training/Validation Accuracy	46
3.13 Neural Network vs Other Models	47

4.1	Partial Autocorrelation with 40 Lags	55
4.2	A Neuron	57
4.3	Multi-Layer ANN	58
4.4	RNN	58
4.5	LSTM	60
4.6	Genetic Algorithm Flowchart	62
4.7	Attributes Correlation	63
4.8	NSW Scatter Plot	65
4.9	SW Scatter Plot	66
4.10	SW Prediction	66
4.11	SW Prediction	67
4.12	SW Prediction	68
4.13	SW Prediction	69
4.14	SW Prediction	70
4.15	Model Predictions Scatter Plot	72
4.16	User's Clusters	74
4.17	Training and Validation Accuracy	75
4.18	Training and Validation Loss	76
4.19	Testing Data Accuracy	77
4.20	Time Taken to Train	78

LIST OF TABLES

Table	Page
2.1 Database Records of a User	14
2.2 Probability of User3 from Location 15	16
2.3 Probability of User3 from Location 67	17
2.4 Top Three Most Probable Locations With Day	24
2.5 Top Three Most Probable Locations With Day and Time	25
3.1 System Configuration	42
3.2 Multiple Linear Regression Model's Configuration	42
3.3 Feature Sets	43
3.4 Hyperparameters of the Classification Model	45
4.1 Different Models and their Characteristics	64
4.2 MSE, R-Squared and Adjusted R-Squared	64
4.3 APAE and Variance of APAE	65
4.4 Hyper-Optimized Models for Currency Exchange Predictions	71
4.5 Model Performance Metrics for Currency Prediction	71

CHAPTER 1

INTRODUCTION

In this chapter, we first introduce the area of this dissertation research, known as Spatial and Temporal Function Estimation (STempFEst), and our research contributions to Human Mobility Function Estimation in Section 1.1. Then, in Section 1.2, we discuss our research contribution to the area of Stocks and Currency Exchange Prediction. Then, in Section 1.3, we give an outline of the remaining chapters of the dissertation.

1.1 Human Mobility Function Estimation

The most utilized application in today's time is the Global Positioning System or GPS. A user through such a device views spatial objects like roads, gas stations, cities or continents. It also helps the user to find different routes to reach a destination dynamically or to look for traffic on a specific route to his/her work place. All such information about the real world data objects, for example, human transit or stationary objects like routes, cities, etc. are stored and retrieved from a database. User's location history is stored every time a user commutes between places and often a user follows a certain pattern in his daily life. According to [1] human behavior is 93% predictable. The research shows that the common perception is that the human's actions are random and unpredictable, human mobility follows a regular pattern.

Most people are equally predictable in their travel patterns even when there lies a significant differences in mobility behavior. An individual's future locations can be predicted based on past travel/location history. We also investigate the high

correlation between time and frequency of travel. The correlation also is between time and locations where the user is currently available. Weather (temperature and precipitation) on the other hand also plays an important role and shows a high positive or negative correlation with frequency of travel.

In this project we implement multiple predictive technologies to predict the future locations of a user. We implement predictive models like

- Hidden Markov Models
- Decision Trees
- K-Nearest Neighbors
- Support Vector Machines
- Feed Forward Neural Networks
- Recurrent Neural Networks (RNN)
- Long Short-Term Memory Neural Network (LSTM)

We begin by implementing Hidden Markov Model (HMM) to predict a user's future locations. Data used in this research was the GPS trajectories collected by Microsoft Research Asia. There are 182 users in a period of three years (April 2007 to August 2012). The dataset is a sequence of time stamped points containing latitude, longitude and altitude [2–4].

This work describes correctly classifying future locations of a user such that we can answer questions like “*Where is the user most likely to be when it is a Monday?*” or day and time-specific queries like “*Where is the user most likely to be between 6:00 pm and 9:00 pm on Saturdays?*”. The first step is to cluster locations and to solve this we use varied K-means algorithm, which unlike traditional K-means, sophisticatedly decides that number of significant clusters of a user. These clusters are then considered as visible states in the HMM and the hidden states are the days of the week and time of the day.

The universal approximation theorem, which states that a feed-forward network with finite number of hidden layers and neurons are capable of estimating continuous functions on compact subsets of euclidean space \mathbb{R}^n [5]. According to this theorem a neural network can represent a wide variety of functions with appropriate parameters (hyper-parameters). Although, the hyper-parameters need to be fine tuned by the network designer.

We then design neural networks to predict future locations both by classifying the clusters and using regression technique to predict coordinates. We fine tune the algorithms by using meta-heuristic technique called genetic algorithm [6, 7]. Meta-heuristics are used to find or generate heuristic designs that provide a sufficiently good solution to an optimization problem especially with limited computational capacity. It samples a set of solutions, which is too large to be completely sampled and make few assumptions about the optimization problem being solved and so they may be usable for a variety of problems. Compared to optimization algorithms and iterative methods meta-heuristics cannot guarantee that a globally optimal solution can be found [8]. In combinatorial optimization by searching over a large set of feasible solutions meta-heuristics can often find good solutions with less computational efforts [9]. Genetic algorithm is inspired by evolution (natural selection, reproduction and survival of the fittest).

We begin by setting the benchmark by implementing traditional machine learning algorithms like, SVM [10], Decision Tree [11], K-Nearest Neighbors [12]. We then implement a regular Feed Forward Neural Network and compare its performance with a Recurrent Neural Network (RNN) and Long Short Term Memory Neural Network (LSTM). All the models are fine tuned using genetics algorithm.

1.2 Stocks and Currency Exchange Prediction

This work describes prediction of purely temporal data using optimized regular and time series neural networks like RNN and LSTM. We shift our domain from spatio-temporal to pure temporal data like stocks and currency exchange price. We gather Apple Inc. stock prices along with other companies stocks, which show either high positive or negative correlation with Apple's stock price. For the purpose of currency exchange price prediction we gather the currency exchange rates between United States Dollar and Indian Rupees. The currency exchange data collected is from 1987 to 2017 (30 years).

Forecasts of stocks can be considered in two categories: technical analysis and fundamental analysis. In technical analysis we only consider past historical data and in fundamental analysis we consider external effects [13]. In our work we have used a hybrid approach that considers both technical and fundamental analysis. We try to predict next day's Apple's stock closing price by including its own high, low, volume, close data along with Microsoft's, IBM's and Standard and Poor's high, low, volume and closing price. We have a total of past 10 years of data from 2008 to 2018.

Prediction of how much a dollar will cost tomorrow can guide one's decision making and can be very important in minimizing risks and maximizing returns. The dataset used in this project is the exchange rate data between January 2, 1980 and August 10, 2017. The dataset displays the value of \$1 in rupees. We have a total of 13,730 records starting from January 2, 1980 to August 10, 2017. Over the period, the price to buy \$1 in rupees has been rising.

We make use of feed forward neural network, recurrent neural network and long short term memory neural network to predict the stocks and currency exchange rates.

1.3 Dissertation Organization

In Chapter 2, we describe the use of Varied-K Means clustering and Hidden Markov Model techniques to predict a user’s future movement based on the user’s past historical data. In this work, we have used real-world geospatial data (including latitude, longitude, day and time) recorded using GPS devices to extract meaningful locations, and model them in such a way that we may predict where a user will be at a given time and day of the week. We cluster the locations using the varied K-means algorithm and implement hidden markov model to learn the pattern of a user not just based on days of the week but also including the time of day. With such a model, we would be able to answer queries like “*Where is a user most likely to be at 6 pm on Wednesday?*”. The visible states display the location clusters of the user but the hidden states show the weekday and time of day. To reduce the number of hidden states, we divided the 24-hours of a day into 8 periods, each period consisting of a 3-hour interval, starting from 12am - 3am, 3am - 6am, ..., 9pm - 12am. In this way, we divided a day into 8 equal periods where each period of a day was a hidden state for each visible state.

In Chapter 3, we implement regression and classification for the purpose of predicting locations by making use of regular feed-forward neural networks. For the purpose of regression the model is successfully able to estimate a user’s spatio-temporal movement function to make accurate predictions and the classifier is able to classify location clusters accurately by learning novel concepts. Our model assumes a completely realistic setting of **first** *the availability of GPS coordinates for individual users*, **second** *an access to open weather API for obtaining current weather conditions*. We have tested the validity of our claims using a real-life dataset from a user and found the accuracy of our model to be accurate 88% of the time.

In Chapter 4 we talk about predicting stocks, currency exchange rates and user's locations using hyper optimized models using genetics algorithm. The models we take into consideration for predictions are feedforward neural networks, recurrent neural networks and long short term memory neural networks.

Finally, Chapter 5 summarizes the contributions made in the dissertation.

CHAPTER 2

Detecting Meaningful Places and Predicting Locations Using Varied K-Means and Hidden Markov Model

2.1 Introduction

This work describes the use of Varied-K Means clustering and Hidden Markov Model techniques to predict a user’s future movement based on the user’s past historical data. Several techniques [14, 15] were proposed to predict a user’s movement, but not many have concentrated on the user’s location based on both weekday and time period within the day. We have introduced a method which models the user’s data, not by just taking day of the week into consideration but also time interval. Our model is able to answer day-specific queries like “*Where is the user most likely to be when it is a Monday?*” or day and time-specific queries like “*Where is the user most likely to be between 6:00 pm and 9:00 pm on Saturdays?*” Our work gives us much higher prediction accuracy than previous research on this topic [16].

In this work, we have used real-world geospatial data (including latitude, longitude, day and time) recorded using GPS devices to extract meaningful locations, and model them in such a way that we may predict where a user will be at a given time and day of the week.

The dataset (GeoLife) contains GPS trajectories collected by Microsoft Research Asia from 182 users during a period of over 3 years. The dataset is a sequence of time stamped points containing latitude, longitude and altitude [2–4]. GPS loggers and GPS phones were used to record the trajectories approximately every 5 seconds. When a user’s location change occurred at a speed of less than 3 mph (assumed walk-

ing speed), the GPS logger did not record the data points. The dataset is distributed over 30 cities in China but mainly comprises locations in Beijing, China. The dataset was cleaned and uploaded to SQLite database.

Since the dataset is just a set of trajectories with a time difference of 5 seconds between pairs of consecutive points (latitude, longitude), there was a need to develop an algorithm which would intelligently group the points together to identify locations where a particular user spent most of their time. These locations are then considered as meaningful places where the user not only frequently went but also where the user spent substantial time.

Though the locations history of a user normally remains private and secured, if a user chose to share their locations with another individual, then an application of this research would be a shared location recommendation system. Consider a scenario where user1 and user2 have shared their locations histories with each other and are aware of each user's normal daily locations and tasks. This information can help them to plan tasks more efficiently and help each other if needed. For example, user1 has their **things to do** "input" list saved on their smartphone and "buying groceries" is listed as a pending task. The system predicts that user2 will be near or at a grocery store during the day while user1 is busy at the office and cannot go. With the help of such a system, user2 would be able to retrieve user1's grocery list and a reminder to buy groceries for user1.

One may also make use of this system in terms of online security. Any website where a user must log in to his/her account using their login credentials needs to place a special focus on securing the user's account on various devices. Such websites make use of the device's MAC address so that when a user tries to log in from any unknown device, the website requires the user to perform an extra step to confirm their identity. If such websites make use of location prediction technology, such as presented in this

paper, then the website will automatically know the predicted behavior of the user and will recognize him/her without having to further confirm their identity.

2.2 Related Works

The two-stage approach to location prediction has been used by several researchers. In [14, 15], the focus is on predicting user's location based on GPS data. They use K-means to cluster the locations, which influenced our approach, but they make use of a basic Markov Model as part of their predictive model. Their prediction is dependent on the user's past location only and does not include day and time as in our work. We use Hidden Markov Model to incorporate weekdays and time as hidden states. Although a person's future location will depend on past and present locations, for a more comprehensive prediction we need to include weekday and time in the features set, which is lacking in [14, 15]. They can predict "*where someone will go next, but not when*". In our research, we can answer the "*when?*" including weekday and time.

[17] has proposed a similar approach by using DBSCAN [18] instead of K-means [19] to cluster the data points, and using a variable order Markov Model instead of HMM for predictions. DBSCAN focuses mainly on the density of the data points and cluster them accordingly. Our K-means clustering algorithm can be customized based on different users and their pattern of stopping at specific locations, which is unrelated to the density of the data points. K-means gave us the liberty to set the number of K clusters, which was calculated by looking at the pattern of "time spent at locations" for each user. Hence, each user has different numbers of clusters based on their own behavioral pattern. Using Hidden Markov Model in our research we included weekday and time in our predictions. In our work, we are successfully able

to answer the questions like “*Where is a user most likely to be at 7pm on Friday?*”. Such queries were not answered in [17].

[20] talks about predicting future locations based on historical data, but also address the “*data sparsity problem*”, which means “*unavailable historical trajectories*”. They propose a method which they call “sub-trajectory synthesis” (SubSyn) to address the data sparsity problem. They also have considered the privacy protection issue in order to hide sensitive location information of a user.

[16] proposed a different technique to cluster user’s data based on temporal characteristics i.e. a cluster for sequences whose visits are made on weekdays by day-time (7am-7pm) and so on. The clustering algorithm is a very simple approximation that uses the timestamp of the last visited location. In our work, we have a more sophisticated algorithm that clusters a user’s data points according to their pattern of stopping at a point of interest.

[21] on the other hand introduces a data mining approach to predict future location of a moving object. The author mines the database of moving object to match the unseen trajectory with the extracted trajectory to select the best association rule.

2.3 Clustering of Locations

We used the K-means algorithm to cluster the dataset but have modified it for our purpose. In general, the traditional K-means algorithm takes randomly-selected-user-defined k numbers of centroids to form k clusters. Each centroid compares its distance from the remaining set of points. The closest points to each centroid make a cluster, for which the mean location is calculated. The mean of all the points in a set is used as the new centroid for the next clustering iteration. This process is repeated until the mean stops changing. Once the mean no longer moves, all the points within

it represent a cluster and are removed from consideration. This process is repeated until no centroid is remaining [19].

Our variation of the K-means algorithm was influenced by [14, 15], which concentrated on "where the user is instead of how the user got there". Our focus was to find locations where a user spent most of their time and to relate those locations to days of the week and times of the day. We targeted our algorithm to find the time elapsed between two consecutive points. If the elapsed time satisfied a threshold, then we marked it as a significant point according to that user. In this approach, we identified the points which have more than a threshold time difference " τ " between them and their corresponding previous points. Another challenge was to find a significant value of τ . In order to do this, we plotted a graph of *Graph to identify meaningful locations*.

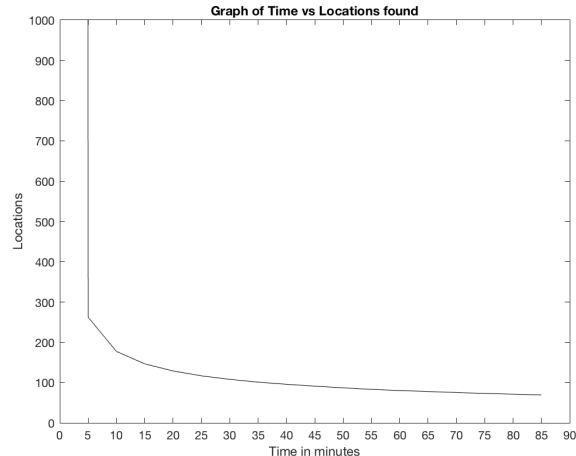


Figure 2.1: Graph to identify meaningful locations.

Figure 2.1 shows the graph that represents the average number of sites (*y-axis*) found for all 178 users when stopped for different durations of time (*x-axis*). We looked for the elbow or knee in the graph and made that duration of time a

generalized condition for all the users on unique locations to focus on where each user stopped. As the time approached zero, the number of sites found were approximately 485,000.

After deciding on right time duration, which in our case was 10 minutes, we started extracting the sites' locations (latitude, longitude). Once we extracted all those points, we kept them in a set which we called the significant sites. In the traditional K-means algorithm, we would need to initialize the algorithm with a value of k . In our approach, the total count of significant sites where a user stopped for at least 10 minutes was chosen as the value of k , or the number of desired clusters.

The next step was to cluster points around these centroids. Since the data set consists of GPS recordings of people from China, the data is spread widely on a city-wide scale. We needed to have a good measure of the radius for a cluster. This was very important because if the radius was too large, we would end up with insignificant places in the cluster which will eventually give us incorrect results. If the radius was too small, we might end up getting one single point in the cluster. To select the best value of the radius for a cluster, we found the distances between each significant centroid. The distance between two points δ was calculated using the Haversine formula, which is defined as:

$$hav\left(\frac{d}{r}\right) = hav(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)hav(\lambda_1 - \lambda_2),$$

where hav is haversine function which is defined as:

$$hav(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \left(\frac{1 - \cos\theta}{2}\right),$$



Figure 2.2: Clusters found for user3 when radius=0.2 miles

d = distance between two points,

r = radius of sphere,

φ_1, φ_2 = latitude of point 1 and point 2,

λ_1, λ_2 = longitude of point 1 and point 2, and

$\frac{d}{r}$ = central angle in radians.

We used the Haversine formula to calculate the great-circle distance between two points on a sphere from their latitudes and longitudes, which is the shortest distance over the earth's surface [22].

After calculating the distances between all sites, we extracted the minimum δ and used it as our radius of the clusters. For different users, δ came out to be different as one value of δ cannot be generalized for all the users. Figure 2.2 displays the clusters for a specific user, user3 in this case. The δ value was set to 0.2 miles.

The black dots are the clusters shown in the figure. There are a total of 253 clusters for user3, 6 of which are shown in the figure. These clusters, if examined closely, are around areas like a residential district, university (Tsinghua University,

Beijing), hotel, airport, etc. A lot of information may be inferred from these clusters for better analysis, and we used such information for our prediction problem.

2.4 Hidden Markov Model

After we ran the clustering algorithm described in the previous section to create clusters of locations based on the two parameters time τ and radius δ , we got a set of clusters which represented the sites or locations where a user tended to visit. After we got all the clusters of locations, we updated our SQLite database where the GPS records are saved chronologically in order to update each user location with its respective cluster id (called LocID). One may name the clusters with specific names (if known) to make more sense of them, like "Home," "Grocery store," "Work place," etc., but we named them with integer ids for our purpose. The database table for a user consists of the records shown in table 2.1:

User_ID	Latitude	Longitude	DateTime	LocID
---------	----------	-----------	----------	-------

Table 2.1: Database Records of a User

Each latitude and longitude point is a member of a cluster and hence gets the cluster id in LocID attribute in the database table. After this process is completed, we have LocIDs in the database updated. These LocIDs are aligned chronologically, which helps us to get the transition from one cluster to another. As an example, figure 2.3 shows transitions between some of the sets of clusters for user3.

A line between any two clusters represents that the user moved from one cluster to another at some point in time. We do not show the direction of transition in figure 2.3, but direction is included in the data itself. We can thus analyze the number of

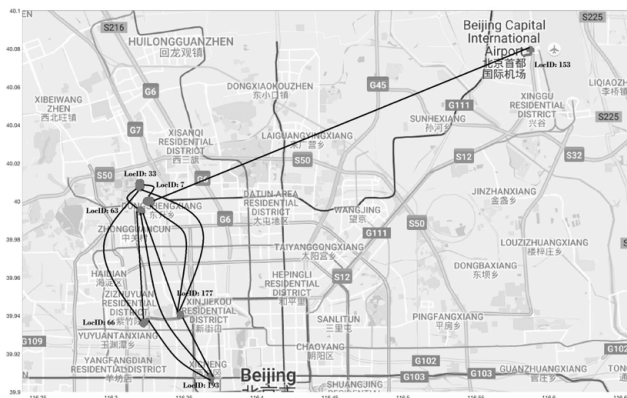


Figure 2.3: Transition Between Clusters

times the user has traveled from one cluster to another. This will help us to calculate the probability of the user travelling from one cluster to another cluster.

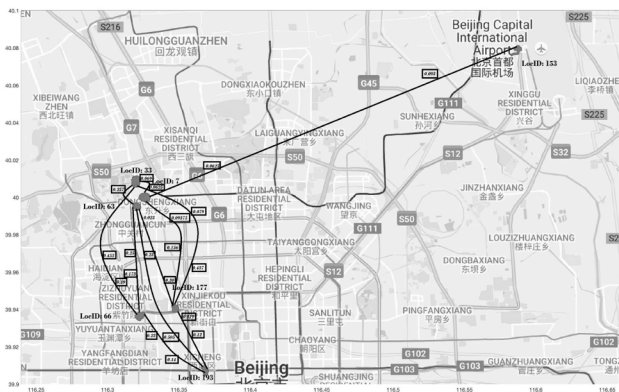


Figure 2.4: Transition Between Clusters with Probabilities

In figure 4, we have shown an example of how the transitions take place between different clusters. Figure 2.4 shows 7 clusters out of 212 clusters of user3. These clusters were given names using integer numbers (1, 2, 3, ..., 212). The ones which are shown in the figure are *LocID 33*, *LocID 7*, *LocID 63*, *LocID 66*, *LocID 177*, *LocID 193* and *LocID 153*. A markov model was created for each location in the map with transitions to other locations. Markov Models are state transition models with the

nodes being the locations with corresponding state transitions between the nodes. It follows the Markov rule which states that the future state depends on the current state and observational data but are independent of past states [23]

Each cluster is a node in the Markov Model and the lines between each cluster represent the probability of the user to transition from one cluster to another cluster. For example, there is a 45% probability for user3 to go to *LocID 66* when the user is currently at *LocID 33* or 23% probability to go to *LocID 193* when the user is currently at *LocID 66*. We ran the algorithm on the whole dataset of user3, which gave us the results in form of transition probability which was used to answer queries like “*Where is user3 most likely to travel next if he is currently at location 46?*”.

Places	Times	Transition	Frequency	Prob.
39	1	15 to 39	1/49	0.02
44	1	15 to 44	1/49	0.02
45	16	15 to 45	16/49	0.33
64	2	15 to 64	2/49	0.04
88	2	15 to 88	2/49	0.04
105	24	15 to 105	24/49	0.48
126	1	15 to 126	1/49	0.02
138	2	15 to 138	2/49	0.04
Total Visits	49			

Table 2.2: Probability of User3 from Location 15

Such results are helpful to analyze the spatial behavior of a user, but with such analysis we miss out on the temporal behavior of the user. The primary objective of this research is to predict a user’s location given a day and a time of the day through which we can answer queries like “*Where is a user most likely to be when it is a*

Places	Times	Transition	Frequency	Prob.
7	1	67 to 7	1/44	0.02
66	31	67 to 66	31/44	0.7
100	3	67 to 100	3/44	0.07
108	1	67 to 108	1/44	0.02
182	2	67 to 182	1/22	0.04
194	2	67 to 194	2/44	0.04
212	4	67 to 212	1/11	0.09
Total Visits	44			

Table 2.3: Probability of User3 from Location 67

Monday?” or *“Where is a user most likely to be when it is 6 pm on Sunday?”* or, *“Give me the next most probable location of a user when he is currently at home and it is Thursday at 5 pm.”* Such queries cannot be obtained by the analysis done above. For such analysis, we make use of the Hidden Markov Model [24] where we introduce day and time as the hidden states for each visible state which, in our experiments, are the site locations.

There are multiple queries one may think of when predicting the location of a user. One of the most common prediction queries is to know where a user is on a specific day like Sunday, for example. To answer such a query, we make use of the Hidden Markov Model represented in figure 2.5, where A, B and C are the visible states, or the clusters, and the hidden states are the days of the week. Thus, by the use of Bayesian approach, we can answer the above query by

$$P(\mathbf{x}|\text{Sunday}) = \frac{P(\text{Sunday}|\mathbf{x}) * P(\mathbf{x})}{P(\text{Sunday})}, \text{ where}$$

\mathbf{x} = ClusterID.

The query result is delivered by finding the \mathbf{x} with the maximum probability. $\mathbf{P}(\mathbf{Sunday} \rightarrow \mathbf{x})$, can be calculated by finding out the total number of visits to cluster \mathbf{x} on **Sunday** divided by the total number of visits to cluster \mathbf{x} . $\mathbf{P}(\mathbf{x})$ is the ratio of total points in cluster \mathbf{x} over the total number of points in all the clusters. Finally, if \mathbf{X} = set of all clusters, then:

$$\mathbf{P}(\mathbf{Sunday}) = [\mathbf{P}(\mathbf{Sunday} \rightarrow \mathbf{x}) * \mathbf{P}(\mathbf{x})] + [\mathbf{P}(\mathbf{Sunday} \rightarrow \mathbf{y}) * \mathbf{P}(\mathbf{y})] + \dots + [\mathbf{P}(\mathbf{Sunday} \rightarrow \mathbf{n}) * \mathbf{P}(\mathbf{n})], \text{ where } (\mathbf{x}, \mathbf{y}, \dots, \mathbf{n}) \in \mathbf{X}.$$

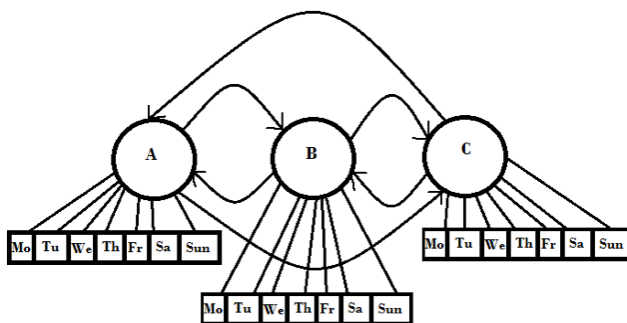


Figure 2.5: Hidden Markov Model for Days In a Week

Next, we take our model to one extra level where we let the model learn the pattern of a user not just based on days of the week but also including the time of day. With such a model, we would be able to answer queries like “*Where is a user most likely to be at 6 pm on Wednesday?*”. Figure 2.6 shows the model where the visible states remain the same as figure 5 but the hidden states have been expanded with the time of day. To reduce the number of hidden states, we divided the 24-hours of a day into 8 periods, each period consisting of a 3-hour interval, starting from 12am - 3am, 3am - 6am, ..., 9pm - 12am. In this way, we divided a day into 8 equal periods where each period of a day was a hidden state for each visible state. As an example, consider figure 6 below.

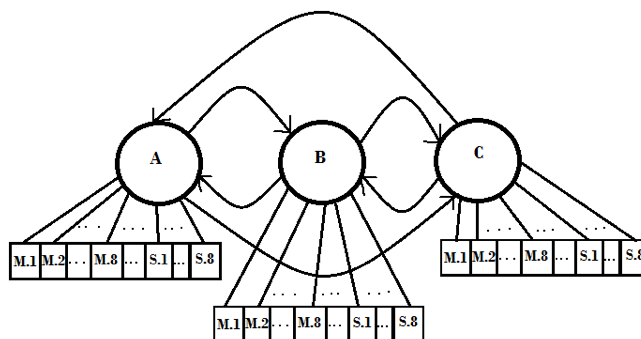


Figure 2.6: Hidden Markov Model for Days and Time

States A, B and C have their corresponding hidden states Monday 1st period, Monday 2nd period, ..., Sunday 7th period and Sunday 8th period. Each period will contribute towards the prediction by having some probability of its occurrence based on the user's historical data. For example, if we were to find the answer for the above query, which asks for the most probable location of a user when it is 6 pm on Wednesday, we will try to answer $\mathbf{P}(\mathbf{x}|\text{Wednesday.6})$, which can be found by calculating:

$$\mathbf{P}(\mathbf{x}|\text{Wednesday.6}) = \frac{\mathbf{P}(\text{Wednesday.6}|\mathbf{x}) * \mathbf{P}(\mathbf{x})}{\mathbf{P}(\text{Wednesday.6})}$$

Even though this looks like a simple conditional probability, if examined closely one may now see an extra feature in our calculations, the addition of time within the day. Intuitively, here we are first trying to calculate the contribution of quarter and day together for a user to be in a specific cluster. For simplicity, we calculate this beforehand by running the algorithm over the whole data set, which we will show in our experiment section, and later plugin the values for the final calculation.

2.5 Experiments

The system used to carry out the experiments had the following configuration: Processor: 2.2 GHz Intel Core i7, Memory: 16 GB, Programming language: MATLAB.

The predictive system we have built in our research is generalized for all the users and the experiments were conducted for 150 users with an average accuracy of 22%. Our accuracy was better than [16], which gave 13.85% prediction accuracy. We attribute this to including day of week and time interval in the HMM, and the use of varied K-means clustering. This is nearly a 70% improvement in prediction accuracy. In this paper we have shown the results for one specific user, i.e user3. user3 has about 500,000 data points.

The first set of experiments was to find the locations where a user is most likely to be on a specific day. Given a cluster \mathbf{x} for user3, we use the following formula (Bayes theorem) to predict where user3 is on Sunday:

$$P(\mathbf{x}|\text{Sunday}) = \frac{P(\text{Sunday}|\mathbf{x}) * P(\mathbf{x})}{P(\text{Sunday})}$$

We find the cluster \mathbf{x} which has the maximum probability. For our experimental purposes, we used the data from multiple users whose data size ranged from 31,830 points to 935,576 points. For the purpose of this paper, we will show experiments for user3 only.

After, we found the clusters of user3, part of which is shown in table 2 and 3, we were interested to see where does this user often go on all the weekdays, i.e., Monday-Sunday. We ran the algorithm and following were the results (table 2.4). We show the top three clusters for each day of the week.

We then matched clusters with known locations on the map. We found places like the university “Tsinghua University, Beijing” and the housing area “Tsinghua Dormitory” and university departments like “Biomedical department” in cluster ID 208 (figure 2.7). This is why the probability of user3 being at cluster ID 208 is highest during the weekdays. The user may be a student, staff or faculty at the Tsinghua University and visits the university during weekdays. If we look at the probability distribution during the weekends, we see different cluster IDs like cluster 195 and cluster 85 having the highest probability.

In this experiment, we have taken a cluster’s radius based on the δ value that was calculated in the previous section of clustering algorithm. Since the dataset is spread in a city-wide scale, and the δ came out to be 0.2 miles, we are predicting a user’s location based on the clusters with size 0.2 miles. There could be another set of places within, say, cluster 208 where the user might be spending more time and which needs to be analyzed. To make more accurate predictions, we would need to cluster places within the clusters found above. This is one of the future works that we intend to work on.

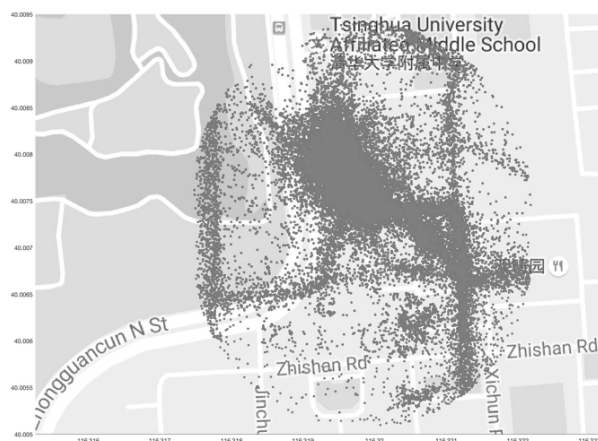


Figure 2.7: Cluster 208

Figure 2.8 displays cluster 195, which has the highest probability on Saturday. If examined closely, we find places like hostels "PekingUni International Hostel" where a user could reside, food places where they might eat and places where a user could go in his/her free time.

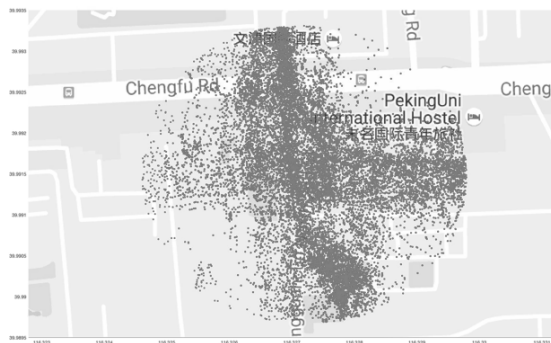


Figure 2.8: Cluster 195

In the second set of experiments, we were interested to find the answer to queries such as *"What is the most likely place for a user to be when it is 6 pm on Sunday?"* To answer this query, we made use of Hidden Markov Model as in figure 2.6. Each hidden state has its corresponding probability of occurrence with respect to the day of the week. We divided each day into 8 equal time intervals or periods. For example, Monday, 8th time interval, which is 9 pm-12 am, has a probability of occurrence for cluster ID, let's say, 208. In such a way, we calculated each interval's probability corresponding to its cluster along with probability of transition from one cluster to another. Once we calculated that, we constructed our Hidden Markov Model which was ready to answer queries like, *"What is the most probable location of a user when it is 2 pm on Monday?"* We ran this algorithm on user3 and found out locations where it is most likely for user3 to be at each period of the day (figure 2.6). Since

the results generated were for each time interval and for each day of the week, for the purpose of this paper (instead of displaying all possible 56 results), in table 2.5, we have shown Monday's 1st, 2nd, 7th and 8th along with Sunday's 1st, 2nd, 7th and 8th period's results.

2.6 Conclusion

In this chapter, we have demonstrated the use of the clustering algorithm K-Means and a predictive technology, the Hidden Markov Model, to predict a user's future locations. We introduced a method which will model the user's data not by just taking day of the week into consideration but also time interval within the day. Our model is able to answer day-specific queries like *"Where is the user most likely to be when it is Monday?"* or day and time specific queries like *"Where is the user most likely to be between 6:00 and 9:00 pm on Saturdays?"* Such a model can be applied to multiple applications which we discussed in the Introduction and Motivation section. We further plan to improve our work with the use of other new technologies which will benefit society in some form.

Monday		Tuesday	
Cluster ID	Probability	Cluster ID	Probability
208	0.211	208	0.238
211	0.155	211	0.128
198	0.064	195	0.061
Wednesday		Thursday	
Cluster ID	Probability	Cluster ID	Probability
208	0.166	208	0.14
211	0.115	211	0.128
195	0.063	195	0.111
Friday		Saturday	
Cluster ID	Probability	Cluster ID	Probability
208	0.156	195	0.194
211	0.101	211	0.088
195	0.084	208	0.0825
Sunday			
Cluster ID	Probability		
85	0.172		
195	0.118		
211	0.092		

Table 2.4: Top Three Most Probable Locations With Day

Monday.1		Monday.2	
Cluster ID	Probability	Cluster ID	Probability
208	0.18241	208	0.18241
211	0.11924	203	0.045506
203	0.045506	51	0.035027
...			
Monday.7		Monday.8	
Cluster ID	Probability	Cluster ID	Probability
208	0.18241	208	0.18241
211	0.11924	211	0.11924
195	0.075195	195	0.075195
...			
Sunday.1		Sunday.2	
Cluster ID	Probability	Cluster ID	Probability
208	0.18241	195	0.1936
211	0.11924	211	0.118
203	0.045506	85	0.0632
...			
Sunday.7		Sunday.8	
Cluster ID	Probability	Cluster ID	Probability
85	0.16245	85	0.15266
195	0.1265	195	0.11767
211	0.09879	211	0.03462

Table 2.5: Top Three Most Probable Locations With Day and Time

CHAPTER 3

Deep Learning Models to Predict User’s Spatial Locations Using GPS Data

3.1 Introduction

Accurate location prediction has been an active objective of consideration in plethora of applications including recommender systems, healthcare applications, traffic planner, marine biology, cellular handshaking, etc. Works abound in literature pertaining location prediction using data mining [25], pattern mining [26, 27], Markov models [28], however application and infrastructure reliance are quintessential constitution of these algorithms (eg: network topology in case of networks [26], radio frequency signal strength reliance [29, 30], specialized hardware [31], social network check-in [32], etc.). We ask ourselves these fundamental questions, ‘what if we do not have the specific information of the infrastructure, can we still make accurate predictions about an individual’s location ?’, ‘How can we learn the features of motion pertaining a particular individual ?’.

Location Prediction using GPS data has sustained its interest to scientists owing to direct applicability in various walks of life. With the prevalence of location identification features in smartphones and wearable accessories, reliable location data became more accessible however, challenges in prediction remained unchanged. Challenges alleviate if we were to account for other factors influencing the prediction such as user preferences, choices and demographic information.

Attributed to a high degree of freedom, modeling human mobility function remains a challenge even for an single user. Even though location identification fea-

tures on smart phones and wearable gadgets enabled faster and reliable GPS data collection, obtaining a dataset fulfilling the research requirements remains a challenge even today. Ascribed to Orwellian implications of social and demographic data pertaining a user, which establishes some causal relations with mobility, individuals remain dubious towards such requests [33] adding further challenges. This work also bridges the gap that ensued from these inadequacies of GPS datasets, by amalgamating a users GPS database with a readily and more openly accessible local weather attributes(using secure APIs), tailoring a *‘unified location prediction model’*.

In a classical study by [14], he conducted experiments to predict future locations using markov model. The dataset Ashbrook used was a single user dataset collected over a period of four months. Inspired by his work we plan to improve the predictions using deep learning on a single user GPS data collected from a student at the University of Texas at Arlington collected over a period of 6 months.

In this chapter we implement regression and classification for the purpose of predicting locations by making use of regular neural networks. As depicted in image we have a representation learning phase and further regression and classification. For the purpose of regression the model is successfully able to estimate a user’s spatio-temporal movement function to make accurate predictions and the classifier is able to classify location clusters accurately by learning novel concepts.

Our model assumes a completely realistic setting of **first** *the availability of GPS coordinates for individual users*, **second** *an access to open weather API for obtaining current weather conditions*. We have tested the validity of our claims using a real-life dataset from a user and found the accuracy of our model to be accurate 88% of the time.

3.2 Related Works

There has been some work done in predicting locations. [14, 15] used two stage approach by clustering user's location using K-Means clustering algorithm and Markov Model as predictive model. They use simple markov models to predict the spatial behavior of a user. In their research, they have not considered time as a dimension, which apparently is one of the important predictive features. Although a person's future location will depend on past and present locations, for a more comprehensive prediction we need to include weekday, time and weather conditions in the features set, which is lacking in [14, 15].

[17] use DBSCAN to cluster the locations and use variable order Markov Model for predictions. We make use of varied K-Means clustering algorithm to cluster a user's data points into significant locations, which can also be customized for different users and stop patterns at different locations. This is unrelated to density of the data points, which is the focus of DBSCAN algorithm.

[34] introduces time prediction methods for buses using multi-layer perceptron. The objective of this work was to predict the real-time travel time of buses. This is however not directly related to our problem but the use of multi-layer perceptron in this work for similar prediction gave us confidence in MLPs for predicting a user's future locations.

[35] propose a new framework for predicting future locations using recurrent neural network. Their model is known as ST-RNN, which stands for spatial-temporal recurrent neural network. ST-RNN models spatial locations and temporal features within each layer to predict where a user is going to be next. The recurrent neural network helps to make intelligent decisions based on the historical pattern followed by the user. However, the system does not account weather conditions, which is one of the most important predictors. Human mobility heavily depends on weather conditions.

According to best of our knowledge no work has included weather information to their dataset to make sensible predictions. Our work demonstrates promising results by having temporal, spatial and weather information in the dataset to make better predictions.

3.3 Methodology

3.3.1 Multiple Linear Regression for Predicting Location Using ANN

One of the methods of predicting “*where is a user most likely to be at when it is Wednesday 6pm, temperature of 70 degrees and precipitation less than 0.1?*” is by implementing multiple linear regression using ANNs. We have different predictive features like time, weekday, month, temperature and precipitation and based on that we try to predict where the user is. The target feature (location) is a linear function of the predictive features. This is also true in general when we say the relationship between features and target is like a straight line or a flat plane and that there is a linear relationship connecting them. In other way to say this is we are formulating a hypothesis,

$$y' = b + X_0.w_0 + X_1.w_1 + \dots + X_n.w_n$$

This hypothesis says y' is the linear function of $X_0, X_1, \dots, X_n \in X$ plus some small error and the parameter b and $w_0, w_1, \dots, w_n \in w$ control our linear hypothesis. [36]

Since this is a supervised learning algorithm the cost function associated with linear regression calculates the residual, i.e., $y_i - y'_i$, where y_i is the ground truth and y'_i is the predicted value. Since the residual carries a sign it will be positive if our hypothesis underestimates the ground truth and negative if hypothesis over estimates it. We can define the total error as the sum of the absolute values of the residuals.

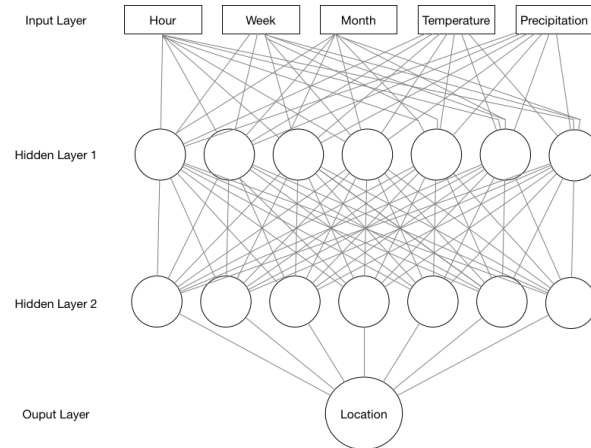


Figure 3.1: Feed Forward Neural Network (Regression)

$$TotalError = \sum_i |\epsilon_i| = \sum_i |y_i - y'_i|$$

The total error is one possible example of the cost function. However, in our work we use mean squared error. Mean Squared Error or MSE is calculated by taking the square of each residual, summing all the squares and dividing by the total number of data points.

$$MSE = \frac{1}{N} \sum_i (y_i - y'_i)^2$$

Since MSE is smooth and is guaranteed to have a global minimum so it is our favorable cost function for this algorithm [37]

Figure 3.1 represents a feed forward neural network or multilayer perceptron. This model is known as feed forward because information flows through the function being evaluated from \mathbf{x} (input layer), through the intermediate computations used to define \mathbf{f} (hidden layer), and finally to the output \mathbf{y}' (output layer). The model does not have a feed back connection through which the output of the model are fed back

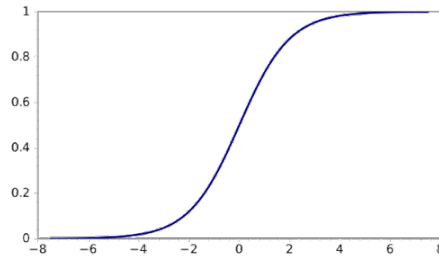


Figure 3.2: Sigmoid Function

into the model itself. By applying a deep feed forward neural network to our problem we can overcome the problem of non linearity.

3.3.2 Location classification using ANN

Amongst the two types of classification, supervised and unsupervised, this section discusses supervised classification. Typical classification technique involves a Logistic Regression; a bivariate classifier. Logistic regression, models the probability of an outcome variable using a logistic curve.

$$y' = f(b + X.w)$$

where the function f is called sigmoid, shown in figure 3.2, and it is expressed by the formula

$$f(z) = \frac{1}{(1 + e^{(-z)})}$$

Unlike in linear regression models, mean squared error could not be used as a cost function in a logistic regression owing to its non-convexity and the ensued inability to find a global minimum. A better cost function to account for error is a log loss or cross entropy cost, defined as follows:

$$c_i = -(1 - y_i) \log(1 - y'_i) - y_i \log(y'_i)$$

where y_i is the predicted probability

$$c_i \begin{cases} -\log(1 - y'_i), y_i = 0 \\ -\log(y'_i), y_i = 1 \end{cases}$$

With the given cost function, we can define a total cost for a single point as the arithmetic mean of the individual errors with the other points.

$$c = \frac{1}{N} \sum_i c_i$$

This cost function is known as average cross entropy or binary log loss. Best use of logistic regression is in binary classifier wherein we have just two classes (0 or 1) [38, 39]. However, our model necessitates a multivariate classification model for classifying entries into multiple clusters, based on varying user locations. The cost function used in this model is called categorical cross entropy. [40]

Softmax imparts probabilities by giving probability distribution for each targets and finds out the class where the probability of the class is maximum.

$$\sigma(Z_j) = \frac{e^{Z_j}}{\sum_{k=1}^K e^{Z_k}}, \text{ for } j = 1, \dots, K$$

Categorical crossentropy is used to calculate loss between two probability distribution by making use of *one hot encoding*.

$$D(y', y) = -\sum y_j \ln y'_j$$

	A	B	C
Location 1	1	0	0
Location 2	1	0	0
Location 3	0	1	0
Location 4	0	0	1

Figure 3.3: Dummy Variables

Neural Networks can be extended to cases where the outputs are not only limited to binary classes. To do regression with multiple outputs, we group the outputs in a single vector and add as many output nodes as the components of the output vector. Each of the nodes will generate an independent output value that will be assigned to the corresponding output vector coordinate.

Classification accuracy is an attribute of the activation function involved and hence, require a careful examination before incorporation into the model. In multivariate classification, we will be predicting Mutually Exclusive Classes or Non-Exclusive Classes. In mutually exclusive classes, an observation can correspond to only one exclusive class and in non-exclusive classes an observation can be a member of multiple classes. A famous example being the document classification. When someone organizes the documents, a document can only be in one single folder, the same document cannot be in any other folder, which is mutually exclusive class. Conversely, we can attach multiple tags to a file, (eg.: a document can be tagged both photo and person, which falls under non-exclusive class). Our research focus is for the mutually exclusive class.

To treat this problem with a neural network, we need to transform the output to a series of dummy binary columns also known as One Hot Encoding. In the case of mutually exclusive classes each row will only host one non-zero value corresponding to the class the record belongs to.

In Figure 3.3 the first row has 1 in column A and 0s in all the other columns and that is because Location 1 belongs to cluster A, similarly Location 3 belongs to cluster B and Location 4 belongs to cluster C. At this point our output is a vector of zeros and ones and we need to apply the correct activation function. In mutual exclusive classes, we want to interpret the output of each node as the probability of being in the corresponding class we need to choose an activation function that forces the sum of the output to be equal to 1. In this way if a location is likely to be in the class ‘A’ it will correspondingly be unlikely to be in any other class. We are achieving this using Softmax activation function.

$$\sigma(Z_j) = \frac{e^{Z_j}}{\sum_{k=1}^K e^{Z_k}} \text{ for } j = 1, \dots, K$$

Softmax is a soft continuous version of the maximum function. Softmax function is a way of forcing the outputs of a neural network to sum to up to one, so as to represent the probability distribution across discrete mutually exclusive alternatives. Softmax receives some total input it has accumulated from the previous layer that is Z_j and the output does not just depend on their own Z_j but depends on the Z 's accumulated by their rivals as well. This way we force the output to lie to represent and a probability distribution over mutually exclusive alternatives.

3.3.3 Optimization Algorithm and Regularization

To train a neural network, we need an algorithm such that we create a facility for the program to be able to understand what it needs to do on its own. The idea is to adjust the learnable parameters, weights (W) and bias (b) values of the neural network based on the cost/loss/error function associated with the neural network. The learnable parameters are used in computing the output values and are learned

and updated in the direction of optimal solution i.e. minimizing the loss. The most common and the widely-used approach to adjust the weights is backpropagation. In backpropagation technique, the network looks at one or a batch of observations, calculates the cost function and based on that adjusts the weights of the neural network by using the concept of stochastic gradient descent (SGD). SGD is a stochastic approximation of the gradient descent optimization method for minimizing the cost function written as a sum of differentiable functions [41].

$$w = w - \eta \nabla Q(w) = w - \eta \sum_{i=1}^n \nabla Q_i(w) / n$$

The stochastic gradient descent applies a correction $\nabla Q(w)$ to the weight w where η is the learning-rate parameter of the SGD algorithm. If smaller learning rate is chosen, changes to the weights are smaller and with larger learning rates larger change to the weights will occur such that the network becomes unstable. We wanted to experiment and observe the changes in accuracy with different learning rates applied to the network, which is currently in progress and kept as one of the future works.

When designing a neural network, which is responsible to estimate a function to represent human mobility, there was a need of generating a complex nonlinear function model [42]. However, overfitting is more likely with nonlinear models that have more flexibility when learning a target function. The model in the case of overfitting picks up random fluctuations and noise in the training data, which impact the model's ability to generalize. To reduce the chances of overfitting we make use of regularization. It is a method to penalize the loss function by adding a multiple of L_1 (Lasso) or an L_2 (Ridge) norm of the weight vector w . In our work, we have used L_1 regularization.

$$L(X, Y) + \lambda N(w)$$

where $N = L_1$ or L_2 and λ is the regularization term.

3.4 Analysis: American Dataset

We validate our claims on improved location prediction of both classification and regression through a field study over a GPS dataset. The data was collected from Google Maps¹ application installed on a mobile phone for a period of 6 months. We would like to emphasis on models veracity for any given user taking an example of a testsubject. Artificial Neural Networks trains itself to accommodate the user mobility preferences amidst high degree of freedom of movement.

The user in our case study was a student at the University of Texas at Arlington, while the data was collected. A smartphone, with google maps installed and location history set to “On”, we collected GPS data from the user for a period of six months (February 2017 - July 2017). During those six months the user mainly browsed various locations around Arlington, Texas. At the end of the sixth month the database collected *9,185 GPS observations*. Each of the tuple recorded was of the form,

$$\langle \textit{Latitude}, \textit{Longitude}, \textit{Hour}, \textit{Day of week}, \textit{Month of year} \rangle$$

with a precision of 7 decimal points for Latitude and Longitude as shown in Figure 3.8, this greater precision in GPS locations helps in achieving greater precision in location prediction.

¹<https://www.google.com/maps>

3.4.1 GPS Data Sampling Rate

As GPS coordinate data is pivotal to our research, the challenge of *determining the appropriate sampling rate* is critical. Figure 3.4 presents the significance of sampling with an interval of one sample per minute. The number of records starts to drop after a three minute difference as you can observe in figure 3.4. This shows that the user moves frequently and in some occasions, stops at some location for greater than three minutes. The user might attend meetings, classes at the university, stops by to eat food or stops at traffic signals. There are some records which have time difference between them and their consecutive points of greater than twenty-five minutes. This is because of the limitation of the battery life. GPS receiver draws about 500 – 600 milliwatts from a smartphone. This is worse for continuous data collection and sometime the phone would die because of low battery. The user missed a couple of hours of data because of this. Another challenge in data collection was low internet signals or resetting the smartphone, which either resulted in no GPS signals or absence of google maps from the phone.

Before discussing the results of the experiments, we would like to highlight some of the observable properties within the GPS dataset which served as impetus to this work.

3.4.2 Hour-of-day - Movement Linking

GPS data is collected using a mobile phone based gps tracking offered by google map based on the given sampling rate. Figure 3.5 illustrates the inherent underlying pattern observed in the dataset, about the user movements and its correlation with the hour-of-the-day. For visualization we have constructed 6, four hour brackets(*12am - 04am, 4am- 08am, 08am-12pm, 12pm-4pm, 4pm-8pm, 8pm-12am*) for movements and clustered points based on its generation timestamp. The histogram shows the

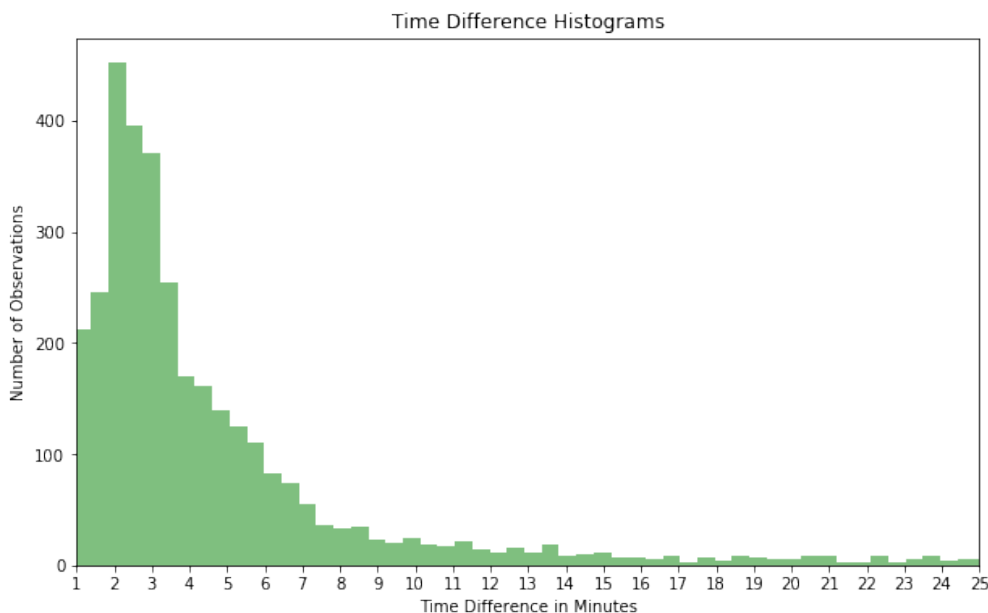


Figure 3.4: Sampling Rate

correlation between the hour of the day and the number of sample points obtained, clustered over different weekdays. We can see a consistent user movement pattern over the graph, however this pattern could not be manifested into locations to qualify as solution points.

3.4.3 Weather and precipitation Correlation with movement

While GPS data does provide some key information pertaining the motion, the datapoints themselves do not provide any cohesive reason for the movement, which is important from a prediction perspective. We have investigated the effects of weather and levels of precipitation on a users movement. We have obtained the weather information on the users location through NOAA(National Oceanic and Atmospheric Administration)²website. Figure 3.6 and figure 3.7 presents the user’s travel behavior during different weather conditions. In Figure 3.6 one can see that how the weather

²<https://www.ncdc.noaa.gov/cdo-web/datasets/LCD/stations/WBAN:53907/detail>

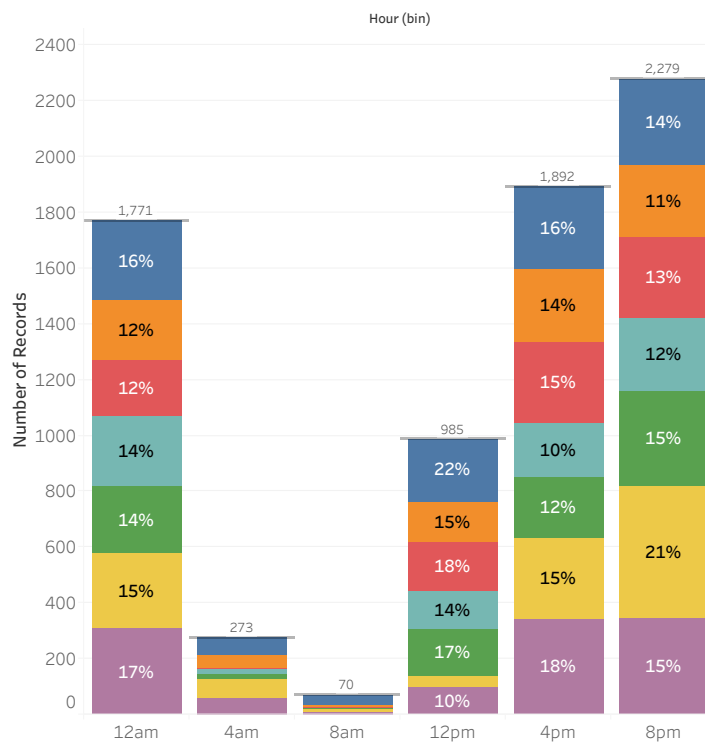


Figure 3.5: Time VS Day

is positively correlated with the number of observations to a certain threshold. The threshold is of 80 degrees or greater. Once the temperature goes beyond 80 degrees the number of observations starts declining. Figure 3.7 has a pure negative correlation between precipitation and number of records. The user during the period of six months did not have a car and used to walk to commute between work, school and home. Whenever the precipitation rises the user tend to commute less.

This kind of analysis confirms that weather plays a very important role in ones commute and the travel plans. We have added the weather features in our dataset which will be very helpful in training the model for predicting the user's locations.

It is clear from the distribution that user avoids extreme weather conditions. The user's movement increases with the increase in temperature up to a certain point (in this case up to 70 degrees). However, the movement drastically decreases after the

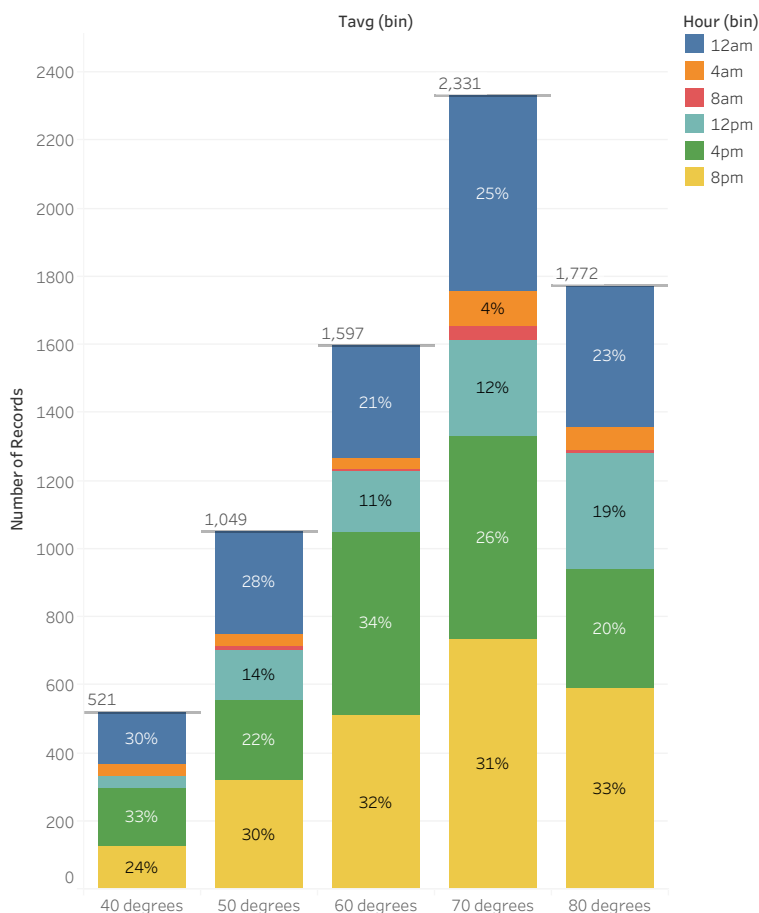


Figure 3.6: Temperature

temperature rises beyond 80 degrees. This explains that the user prefers temperature between 70-80 degrees and prefers to travel most between then. This shows that temperature plays an important role in one's travel decision making

The map in figure 3.8 displays all the locations from the user's historical GPS dataset. The color bar shows different months starting from February 2017 (2) to July 2017 (7). One can see that there is one color trajectory (April 2017) represented as 4 starting from Dallas-Fort Worth (DFW) region to Houston, Texas. This is because the user had to attend a conference for one day in Houston in the month of April. However, DFW region is covered with multiple colors as shown in figure 3.9, this

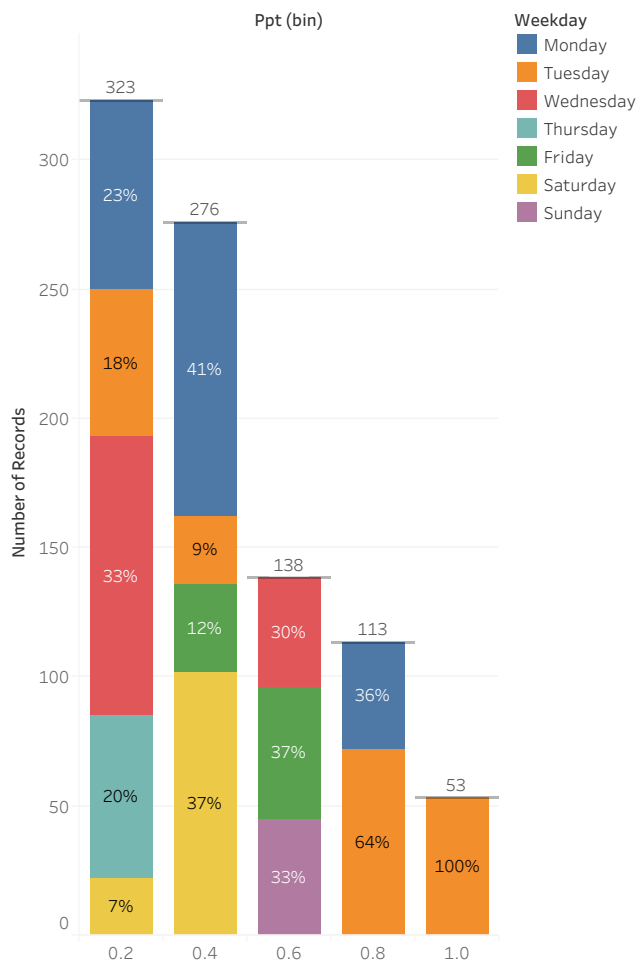


Figure 3.7: precipitation

explains user's high movement around DFW area. Just by looking at the map one cannot predict where and what time a user is most probable to travel in future. To answer such questions, we need a sophisticated model which can take all the predictive features into account to answer the user's most probable future locations.

3.5 Experiments

The system used to carry out the experiments had the configuration shown in table 3.1

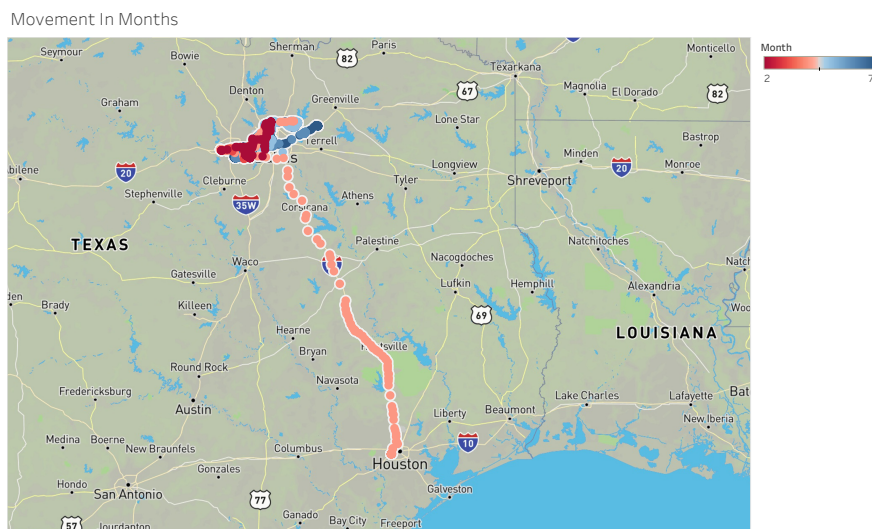


Figure 3.8: User Locations for 6 Months

Processor	Memory	Language	Package
2.2 GHz Intel Core i7	16 GB	Python 2.7	TensorFlow/Keras

Table 3.1: System Configuration

3.5.0.1 Multiple Linear Regression Using Neural Network

The model used for the purpose of multiple linear regression had the configuration shown in table 3.2

For regression we carried out two different sets of experiments. We wanted to see how the results are affected if we just use following as feature set, which is explained in table 3.3.

Layers	Neurons	Activation Function	Cost	Optimizer
Hidden Layer 1	6 neurons	Tangential Hyperbolic		
Hidden Layer 2	4 neurons	Tangential Hyperbolic		
Hidden Layer 3	3 neurons	Tangential Hyperbolic		
Output Layer	2 neurons	Pure Linear	MSE	SGD

Table 3.2: Multiple Linear Regression Model's Configuration

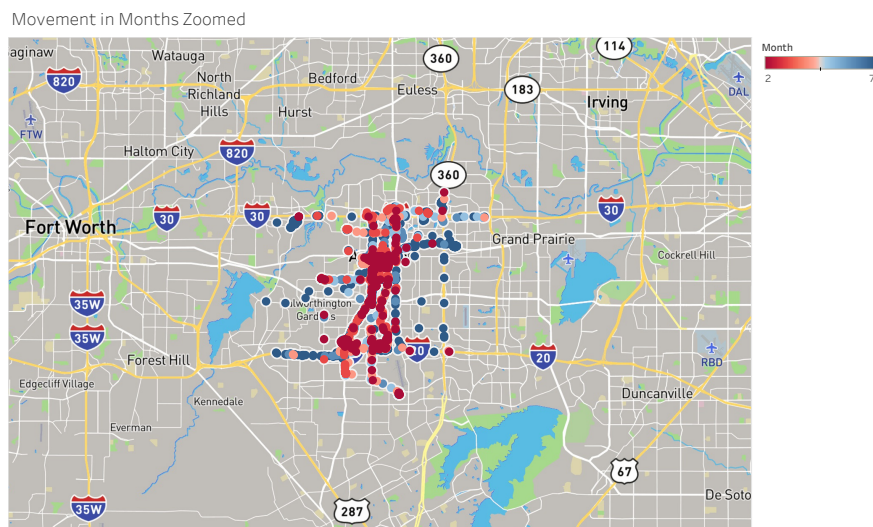


Figure 3.9: Frequent user Locations for 6 Months

We calculated model's loss (shown in figure 3.10) and validation loss (shown in figure 3.11) while training the model. For the purpose of validation loss we took 20% of the training data as the validation data.

From figure 3.10 and figure 3.11 it is clear that the model predicts more accurate locations with less loss when we add temperature and precipitation in the feature set.

3.5.0.2 Classification Using Neural Network

The model used for the purpose of classification had the hyperparameters listed in table 3.4.

After learning that weather features play an important role in predicting a user's location we carried our classification experiments with weather features appended to

	X_1	X_2	X_3	X_4	X_5	Y_1	Y_2
Set 1	Month	Weekday	Hour	–	–	Lat	Lon
Set 2	Month	Weekday	Hour	Temperature	Precipitation	Lat	Lon

Table 3.3: Feature Sets

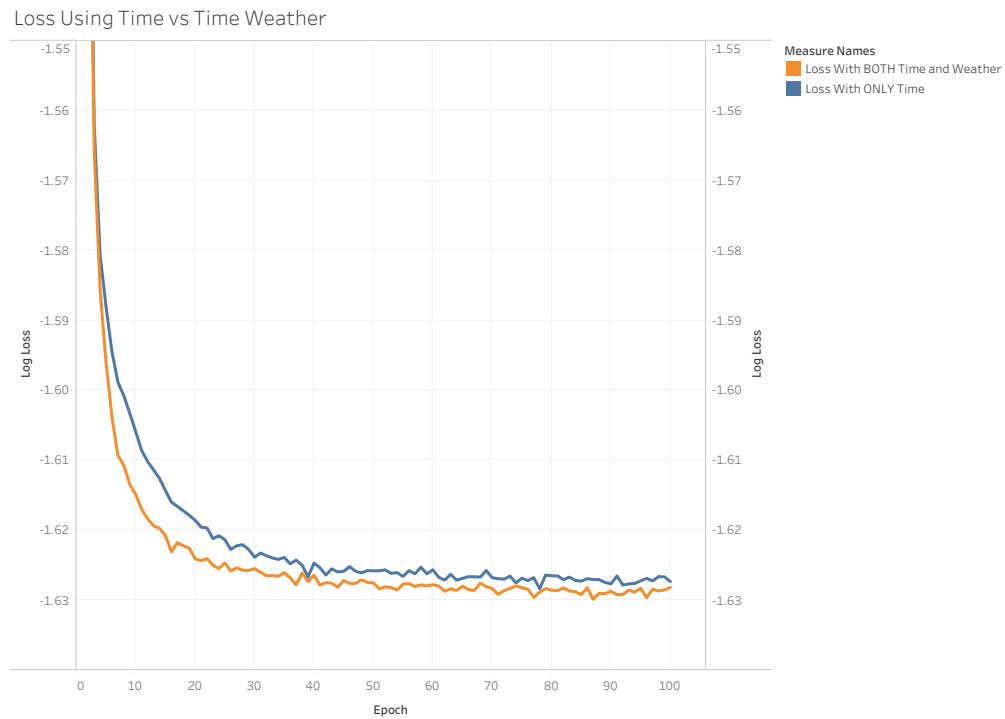


Figure 3.10: Training MSE

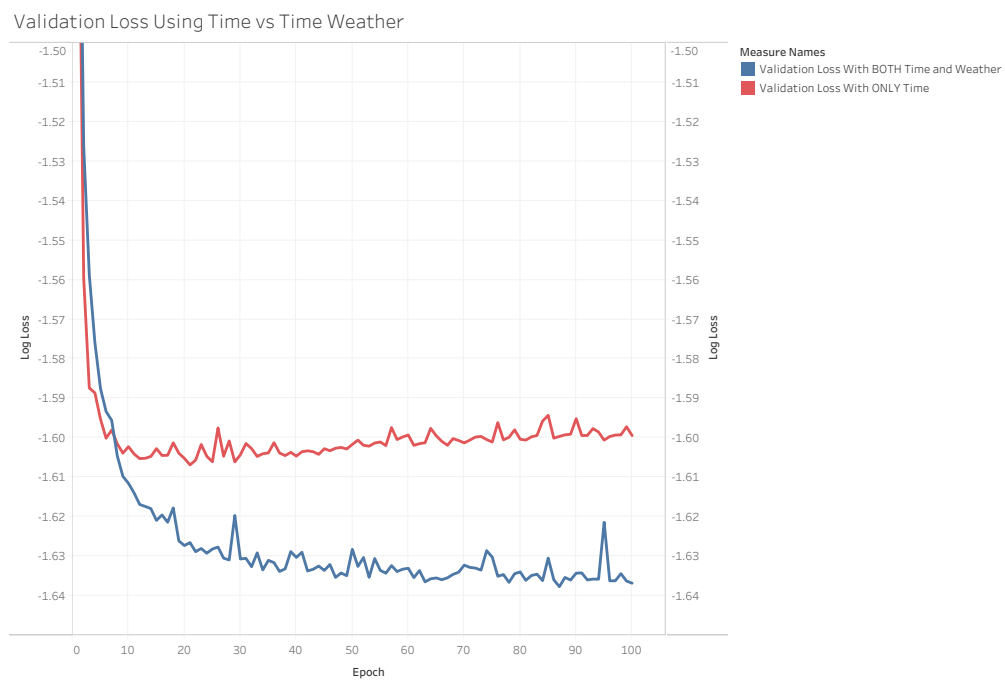


Figure 3.11: Validation MSE

Layers	Neurons	Activation Function	Cost	Optimizer
Hidden Layer 1	5 neurons	Tangential Hyperbolic		
Hidden Layer 2	5 neurons	Tangential Hyperbolic		
Hidden Layer 3	5 neurons	Tangential Hyperbolic		
Output Layer	Total Clusters	SoftMax	CatCross	SGD

Table 3.4: Hyperparameters of the Classification Model

the dataset. We used varied K-Means clustering algorithm from the previous chapter 2 to cluster the locations in American study. There were a total of 9 clusters found in the user's data, which we used as classes for the purpose of classification of locations.

The model is able to generalize well and that is shown by the validation accuracy in figure 3.12.

After training the model we also compared our Neural Network with other traditional classification models by using an average of 3-fold cross validation check and clearly our Neural Network is much more efficient as compared to traditional classification machine learning algorithms like K-nearest neighbors, SVM and Random Forest classifier, which is shown in figure 3.13. In average our Neural Network performed 1.41 times better than any of the mentioned traditional machine learning algorithms to classify the right cluster of the user.

3.6 Conclusion and Future Work

Till now we have made three models that can help one to predict a user's location based on time and weather information. We talked about Hidden Markov Model in chapter 2 and however HMMs are powerful to learn the probability distribution of different states and model time-varying spectral vector sequence, hidden markov model has poor accuracy and unacceptable sensitivity to changes in operating environment. Considering this aspect we decided to implement neural networks

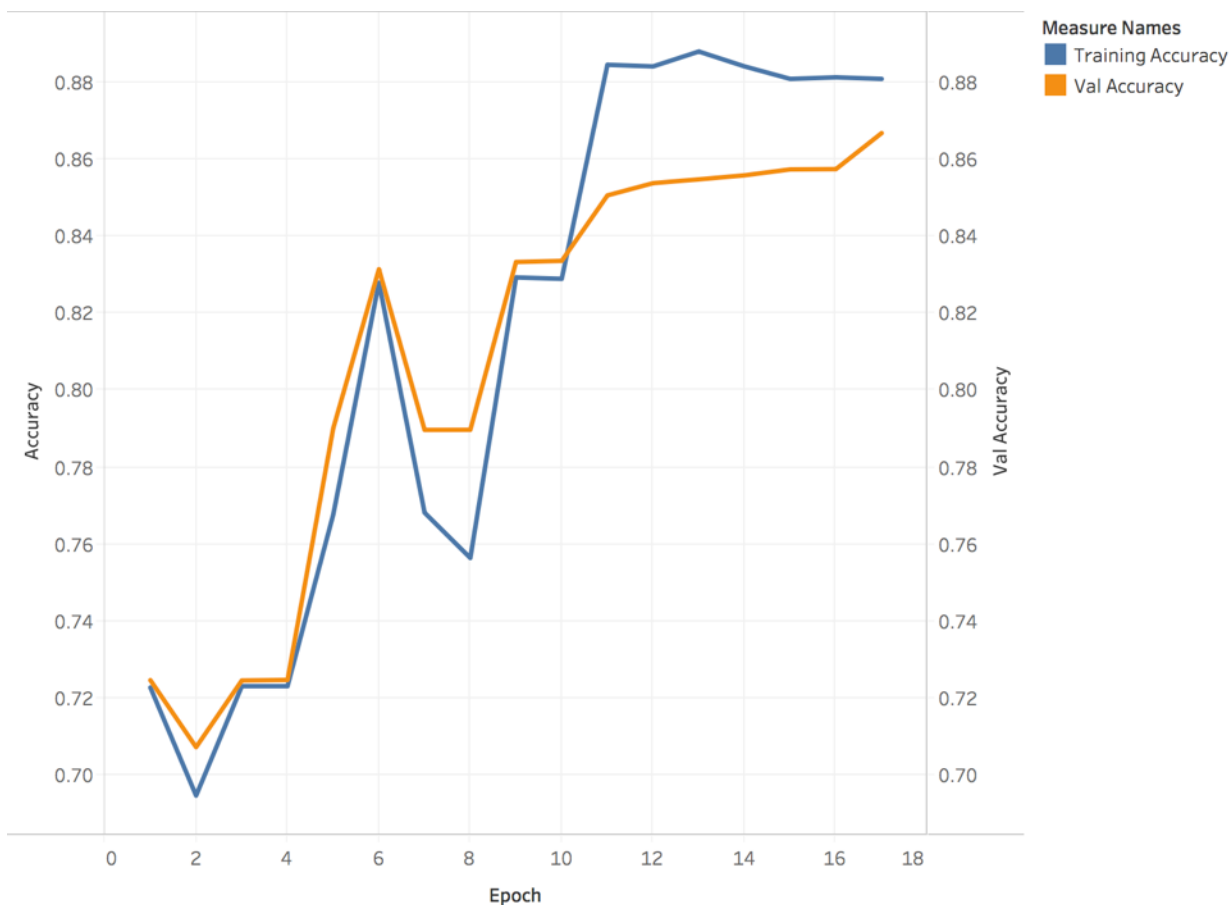


Figure 3.12: Training/Validation Accuracy

which are more open to changes in the environment. We compared the performance of the neural network by comparing it with other traditional algorithms and clearly neural networks stand at the top in the case of classification and regression. In the next chapter we incorporate a time series model or a recurrent neural network. Long Short Term Memory (LSTM) Neural Networks are the best example of a recurrent neural network. LSTM are different from a regular recurrent neural networks because a simple recurrent neural network suffers from a fundamental problem of not being able to capture long-term dependencies in a sequence. This is a problem because we

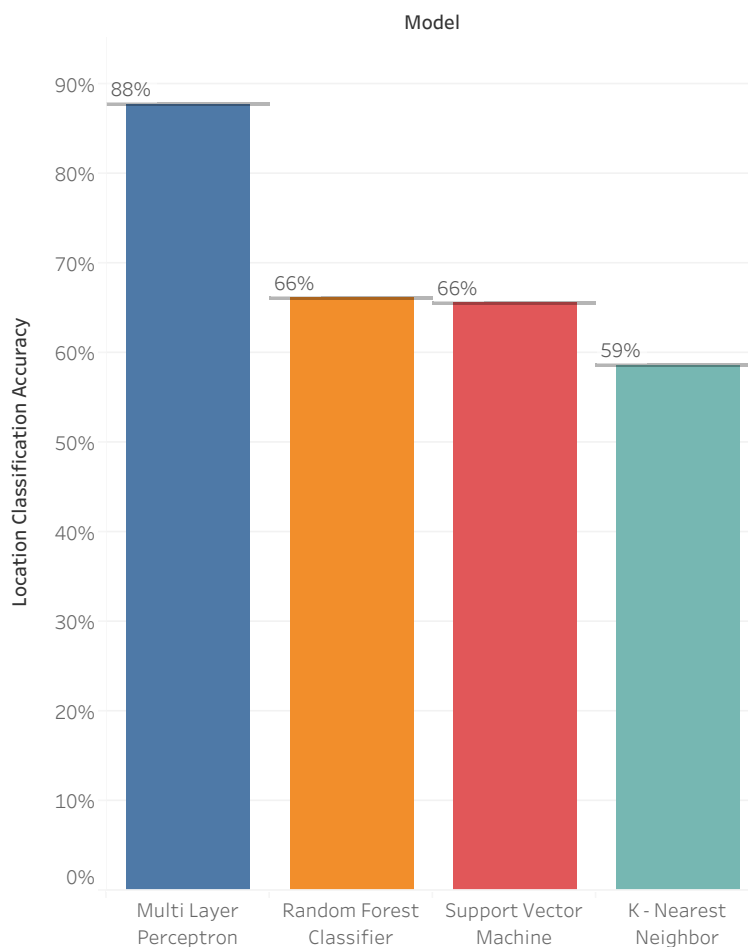


Figure 3.13: Neural Network vs Other Models

want our RNNs to analyze data and answer questions, which involves keeping track of long sequences of locations [43].

However, setting up a neural network requires one to set up different hyperparameters like, number of hidden layers, number of neurons or nodes within each layer, activation functions, cost function, optimizers etc. This problem falls under the umbrella of “Tuning Hyperparameters”. One way to find the right hyperparameters is through brute force trial and error method. However just to set up the right hyperparameters for a model which has four parameters with five possible settings each will take about $(5^{**4}) * 5$ minutes, which is equal 3,125 minutes or about 52 hours.

To overcome this we use “Genetic algorithms”. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as selection, crossover and mutation. Through this we plan to make a system which has intelligence of a human and computation power of a machine. Such systems have already shown their performance and eligibility to overcome any problem with accurate results.

CHAPTER 4

Hyper Optimized ANNs, RNNs and LSTMs with Sliding Window to Predict Short Term Apple's Stocks Prices, Currency Exchange and GPS Locations

4.1 Introduction and Related Works

Predicting stocks prices for different companies has been the most challenging and intriguing tasks in the field of data science. Researchers have tried to solve the problem by various means including and not limited to data mining [44], traditional machine learning [45, 46], neural networks [47], time series neural networks [48, 49] and other machine learning methods. Predicting next day's/week's/month's/year's stock return is a time series problem with time as an important dimension, which is solved by an algorithm by estimating the function of the moving stocks. Machine learning models are mathematical formulas with a number of parameters, which are learned from the data. Models that have been built till now to estimate this time series functions have learned parameters from the historical data presented that were shown to the models. Hyperparameters are another kind of parameters, which cannot be learned directly from the regular training process. The practitioner sets the hyperparameters before the training process begins. Hyperparameters on a higher level defines model's complexity and ability to learn complex non-linear functions. These are not learned by showing the data to the model but instead are set by a practitioner by test and trial methods and opting the one, which gives the best results. On the other hand, hyperparameter optimization techniques are used to set

the right hyperparameters. It is a task of finding an optimal or near-optimal set of hyperparameters.

Till now researchers have done extensive work in creating models to predict stock prices but to the best of our knowledge most of the works have not shown techniques to optimize the hyperparameters and find “the best” model out of all the possible population of models to predict stock prices as close as possible. In this work we have worked on a meta-heuristic hyperparameter optimization technique called genetics algorithm through which we have selected the best model for specific kind of problem. We also make use of sliding window technique to capture the patterns within the data for better prediction. Some research also used the sliding window technique [48, 50] and decided the window size based on error for various window sizes. We on the other hand have decided the size of window more sophisticatedly where we calculate the partial-auto-correlation between the data to calculate the best window size. Partial-autocorrelation function measures the correlation between an observation with another observation, which is n time steps apart.

Forecasts of stocks can be considered in two categories namely: technical analysis and fundamental analysis. If the forecast is dependent only on historical data such as past values, volume of trading etc. then it is a technical analysis. On the other hand, if the forecast is based on external effects such as currency exchange rates, interest rates etc. then it is fundamental analysis [13]. In our work we have a combination of technical and fundamental analysis. We will see in coming sections that how these both play a major role and what all features are important in predicting Apple’s stock prices.

There have been a lot of work done where some researchers [46] tried to assess different function estimating models while comparing with other models. Some work also compared traditional machine learning models with artificial neural networks [45].

By making use of hyperparameter optimization we try to compare different artificial neural networks among themselves and try to find the best model with the right hyperparameters for a certain kind of problem. [51] tries to compare different types of neural networks but is missing the hyperparameter optimization in their models.

[44] uses sentiment analysis techniques to construct a sentiment space to calculate news impact on stocks and use them as a feature set in predicting stock prices. Their intention is to correlate the market behavior with financial news. They believe that financial news plays an important role on stock price return. In this work they have analyzed the news impact from sentiment dimension.

[45] on the other hand have proposed ensemble methods in machine learning to predict stocks. They have fused models like Support Vector Regression, Random Forest Regressors and Artificial Neural Networks to create SVR-ANN, SVR-RF and SVR-SVR predictive models. In this work the researchers have tried to predict short term stock prices 1-10 days and relatively long term prices 15 and 30 days by giving the $(t)^{\text{th}}$ day values as inputs to predict $(t+n)^{\text{th}}$ day value (where t is the current day). This work shows that the two-stage predictive models outperform the single-stage predictive models and that the best overall prediction was achieved by SVR-ANN model.

Another example of ensemble methods can be seen in [46] work. In this work they have benchmarked ensemble methods (Random Forest, AdaBoost and Kernel Factory) against single models such as Neural Networks, Logistic Regression, Support Vector Machines and K-Nearest Neighbors classifiers. By making use of above seven different predictive modeling techniques they setup a benchmark in the field of stock prices forecasting. In this work the predictions are one year ahead so all the algorithms are used for a relative long-term forecasting. The main focus of this work is not to predict absolute stock prices but the direction of stock prices hence this is a

classification problem instead of a regression technique that we have our work focused on.

[47] make use of artificial neural networks and principal component analysis to predict the stock prices. In this work the researchers have made use of principal component analysis technique to find the most important components that shows high variance across the dataset and have made use of artificial neural networks to predict the stock prices.

Comparison of different types of neural networks in predicting stock prices have been done in [13]. In their work the researchers have taken MLP, RNN and CNN as three different models and have compared their performance in predicting stock prices. All the models have been optimized to get right and optimum weights, dropout rates and even the kernel initializer techniques using Tree-structured Parzen Estimator (TPE) [52]. Their work showed that CNN outperformed the rest two models in predicting the financial data. They also mentioned that how ensemble methods in neural networks can outperform any single model not only in just financial data prediction but also in classification tasks like Netflix customer classification.

[51] have compared three different types of neural networks namely, Multi-Linear Perceptron, Dynamic Artificial Neural Networks and Hybrid Neural Networks which use Generalized Autoregressive Conditional Heteroscedasticity. In their work they determined that MLP outperformed other two neural networks and even a hybrid model GARCH-ANN did not give satisfying results. GARCH-DAN2 hybrid model had the worst result while MLP outperformed GARCH-MLP with a little difference. Their work and experiments showed an important aspect of comparison between different neural networks but if the model's hyperparameters were optimized based on the problem they were required to solve the results could have been different.

[53] use Singular Spectrum Analysis (SSA) to analyze and understand the non-linearity and non-stationarity in a time series data. The purpose of SSA is to extract series of singular values that explains the original series using Singular Spectral Decomposition (SVD). The purpose of this method is to extract and reduce noise in time series. Once they decompose the time series using SSA into trend, market fluctuation and noise they then use Support Vector Machine (SVM) to make price predictions. They also use method like Ensemble Empirical Mode Decomposition (EEMD) to extract fundamental features in the time series and provide to SVM for predictions. Their findings show that SSA-SVM outperformed EEMD-SVM predictions and that SSA extracts features better than EEMD.

There are other more sophisticated types of neural networks that have been extensively used for forecasting and predicting time series data. Such networks are known as Recurrent Neural Networks (RNN).

[50] proposed a new hybrid model known as Hybrid Prediction Model, which is a combination of RNN, Exponential Smoothing Model (ES) and Autoregressive Moving Average Model (ARMA). In this work they generate input-output pairs using Autoregressive Moving Reference (ARMR), which is then fed in RNN in a supervised fashion (AR-MRNN). HPM is constructed by merging the predictions made by RNN, ES and ARMA. They claim that RNN makes better predictions than a linear model but HPM outperformed RNN and that the HPM uses Genetic Algorithm to set the optimum weights.

[49] in their work have used Long Short-Term Memory, which is another kind of recurrent neural network to predict stock prices. LSTM is widely known for its capability to process data which has time as a dimension associated with it. Example data like sounds, videos or even stock prices. They have used LSTM models to predict China stock return and claimed to improve the accuracy of stock returns prediction

from 14.3% to 27.2%. In their work they presented the dataset to the model in 5 different ways where each dataset was feature engineered in some way. They presented the data by adding closing and volume data, normalized closing and volume, adding high, low, open to the data, adding close and finally adding volume. They recorded constant growth in accuracy by adding more dimensions to the dataset.

[48] on the other hand made use of LSTM but also incorporated regular RNN and Convolutional Neural Network (CNN) by making use of sliding window. In this work the researchers have introduced a lag in the dataset by using sliding window and shown this lagged dataset to LSTM, RNN and CNN. They choose the size of a sliding window, which gives the least error. They have also benchmarked the research by making use ARIMA model to predict the stock prices and clearly the neural networks have outperformed the ARIMA model. The models are quantified using percentage error. LSTM and RNN models are outperformed by CNN due to the sudden changes that occur in stock markets.

4.2 Methodology for constructing and optimizing models

4.2.1 Four Main Characteristics of Each Experiments

As explained before, forecasts of stocks can be considered in two categories: technical analysis and fundamental analysis. In technical analysis we only consider past historical data and in fundamental analysis we consider external effects [13]. In our work we have used a hybrid approach that considers both technical and fundamental analysis. We try to predict next day's Apple's stock closing price by including its own high, low, volume, close data along with Microsoft's, IBM's and Standard and Poor's high, low, volume and closing price. We have a total of past 10 years of data from 2008 to 2018. Our ongoing work plans to include every day's Apple's sales data

along with currency exchange rates between US Dollars and Indian Rupees, Chinese Yuan, Japanese Yen and Euro data in the analysis. There are a lot of unknown features, which are important in predicting stock price. We try to add as many features and based on their coefficients and p-values we identify the statistically significant ones and eliminate the others.

This work is based on sliding and non-sliding window approach. Our work is divided in two major categories of experiments where we compare the performance of different predictive models when presented with non-sliding window data versus sliding window data. The window size is set by calculating the partial autocorrelation between the stock prices for lags ranging between past 10 through 40 days and the best window size was last 30 days of window. The partial autocorrelation shown in figure 4.1 explains that there were either strong positive or negative correlation between values of past 30 days and the correlation gradually converged to zero as we moved further past 30 days.

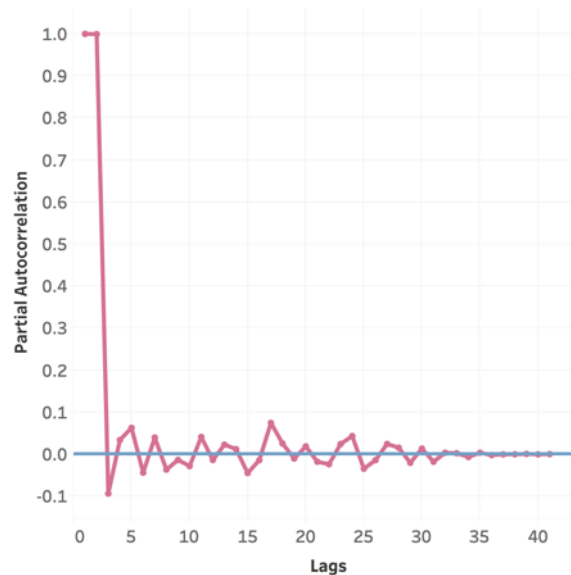


Figure 4.1: Partial Autocorrelation with 40 Lags

We present the data to models through Non-Sliding Window (NSW) and Sliding Window (SW) techniques. For each technique we make use of three different neural network models namely Artificial Neural Network (ANN), Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Neural Network. Model's hyperparameters are optimized by making use metaheuristic optimization technique called Genetic Algorithm [6, 7].

4.2.2 Artificial Neural Network

Artificial Neural Network is composed by nodes (Neurons) and each node can have many inputs (X_1, X_2, \dots, X_n) but will only have one output (next day's stock price). As shown in figure 4.2, a neuron is connected to input X_i by weight W_i . A second edge enters the node carrying the value b , called a bias. The net output n is calculated by $\sum_{i=1}^n W_i \cdot X_i + b$, which is passed through an activation function to make the neural network non-linear. To make the network capable of answering complex queries and find non-linearity we extend this network with more neurons and more hidden layers as shown in figure 4.3. The first layer computes $Z_1 = X \cdot W_1 + B_1$, where X, W, B are vectors of inputs, weights and biases respectively with as many components as the number of nodes in the layer. The output of the layer will be the activation function applied to the linear transformation $O_1 = f(Z_1)$. This output is fed to the connecting layer and same computations are done for the following layers. This structure allows to introduce very complex non-linear relationships between input and output values.

We have historical stock price data with next day's stock price as the dependent variable. We train our ANN in a supervised manner where the cost function associate with linear regression calculates the residual, $y_i - \hat{y}_i$. In our work we use mean squared error because $MSE = \frac{1}{N} \sum_i (y_i - \hat{y}_i)^2$ is smooth and is guaranteed to have a global

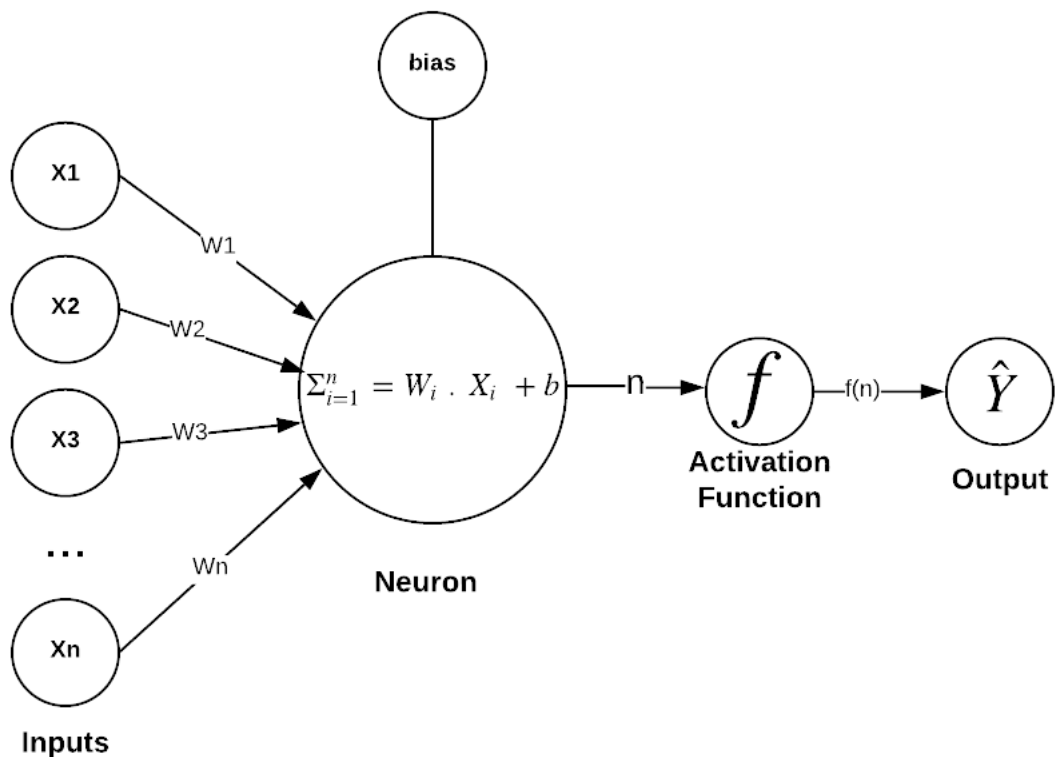


Figure 4.2: A Neuron

minimum. We tune the weights of the network using Adam optimizer. The authors [54] claim Adam to be fast in achieving good results by making use of exponential moving average of gradient and the squared gradient where the parameters β_1 and β_2 control the decay rates of the moving averages.

4.2.3 Recurrent Neural Network

Unlike ANNs, which we described in previous section RNNs have hidden states distributed across time, which make them to store important information about the past. Unlike ANNS, RNNs are dynamic neural networks because the output depends on the current input as well the past hidden states of the network. As shown in

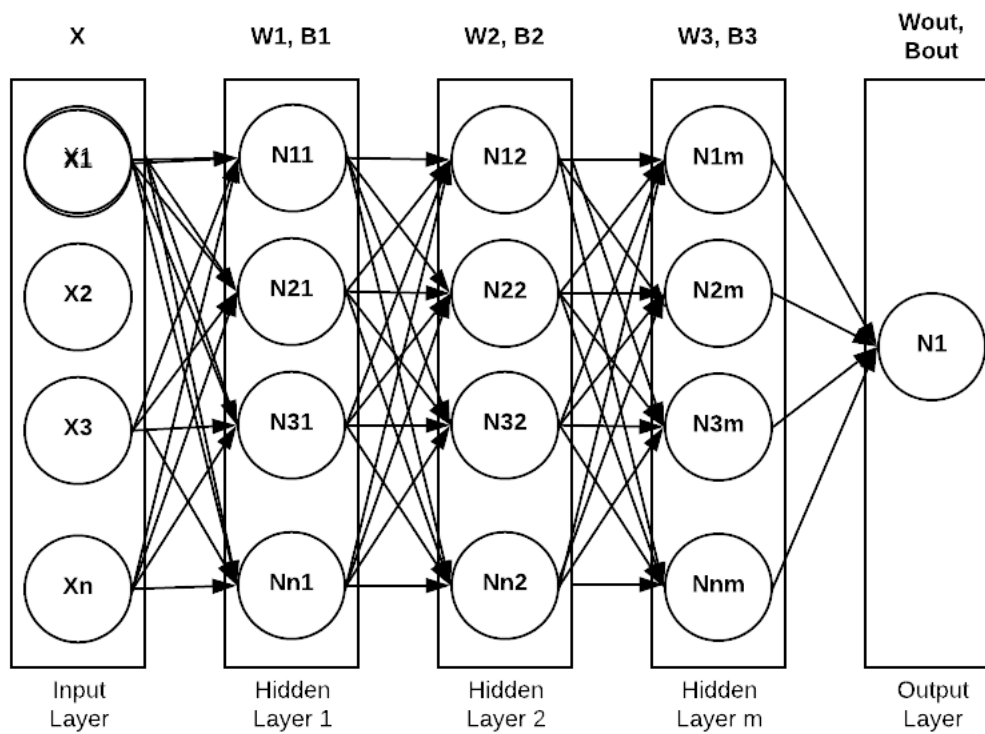


Figure 4.3: Multi-Layer ANN

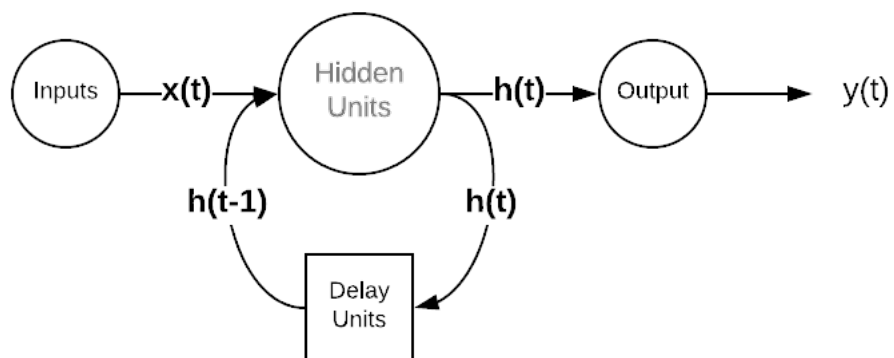


Figure 4.4: RNN

figure 4.4, at time step t the model processes the input vector $x(t)$, calculates the hidden state $h(t)$ eq. 4.1 using an activation function to predict the output $y(t)$ eq. 4.2. At each forward pass the model holds or remembers the hidden state in the delay unit, which is fed back to the hidden layer as additional inputs.

$$h(t) = ReLU(W_h^T \cdot h(t-1) + W_x^T x(t) + b_h) \quad (4.1)$$

$$y(t) = ReLU(W_o^T \cdot h(t) + b_o) \quad (4.2)$$

Since the feedback connection represents a time delay of one so the hidden layer takes both input vector at time t and hidden state at time $t-1$. The activation function we choose is ReLU function as it can help with vanishing gradient problem [55]. Recurrent neural network suffers from a fundamental problem of not being able to capture long-term dependencies in a sequence. This is a problem because we want our RNN to analyze the stock prices, which involve keeping track of long sequences of past stock prices.

4.2.4 Long Short-Term Memory Neural Network

Long Short-Term Memory (LSTM) was proposed by [43] in late 90's. LSTM is a special kind of RNN where hidden state from the RNN is replaced by the memory block or LSTM cell. Figure 4.5 shows a basic memory block of LSTM, which includes several operations performed by forget gate, input gate and output gate. We describe function of each gate below.

1. Forget Gate: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

The forget gate is a sigmoid layer, which ranges between 0 and 1. It helps to decide whether the previous internal state is completely forgotten or is passed through unaltered.

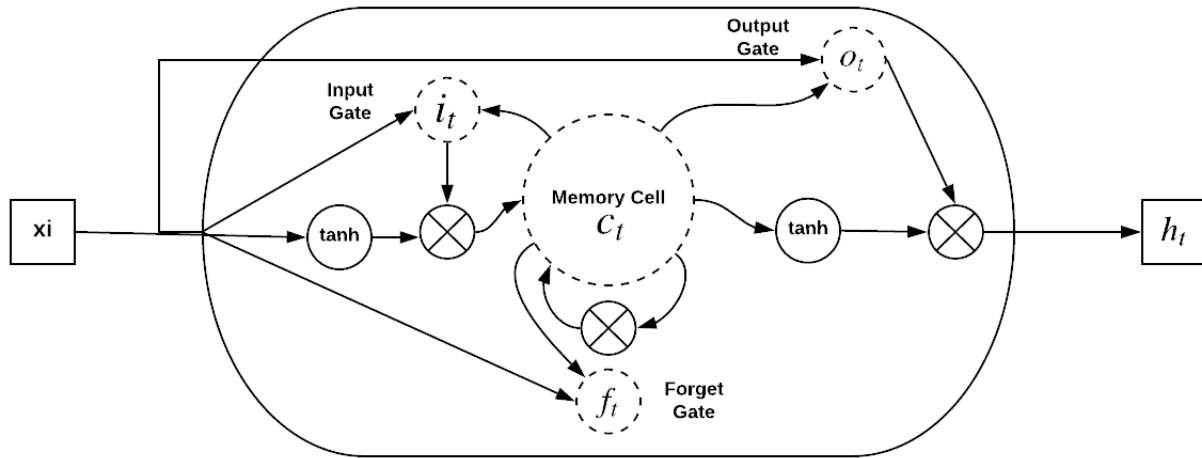


Figure 4.5: LSTM

2. Input Gate: $i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$

The input gate takes the previous output and the new input and passes them through another sigmoid layer. The value of the input gate is multiplied with the output of the candidate layer C_t .

$$C_t = \tanh(W_c.[h_{t-1}, x_t] + b_c)$$

This layer applies a hyperbolic tangent to the mix of input and previous output, returning a candidate vector to be added to the internal state. The internal state is updated with this rule:

$$C_t = f_t * C_{t-1} + i_t * C_t$$

The previous state is multiplied by the forget gate and then added to the fraction of the new candidate allowed by the output gate.

3. Output Gate: $O_t = \sigma(W_o.[h_{t-1}, x_t] + b_o; h_t = O_t * \tanh(C_t)$

The output gate controls how much of the internal state is passed to the output and it works in a similar way to the other gates.

4.2.5 Hyper-Parameters Optimization

We make use of meta-heuristic technique called genetics algorithm to optimize the hyperparameters of our three different predictive models. The hyperparameters we try to optimize are: hidden layer size, neurons in each layer and weight initialization. Meta-heuristics are used to find or generate heuristic designs that provide a sufficiently good solution to an optimization problem especially with limited computational capacity. It samples a set of solutions, which is too large to be completely sampled and make few assumptions about the optimization problem being solved and so they may be usable for a variety of problems. Compared to optimization algorithms and iterative methods meta-heuristics cannot guarantee that a globally optimal solution can be found [8]. In combinatorial optimization by searching over a large set of feasible solutions meta-heuristics can often find good solutions with less computational efforts [9]. Genetic algorithm is inspired by evolution (natural selection, reproduction and survival of the fittest). The entire process can be represented as a flowchart shown in figure 4.6.

We apply this algorithm for all the three models to get the optimized number of hidden layers, number of neurons for each hidden layer and weight initialization. We keep optimizer, learning rate, activation function, regularizer and number of epochs constant for all the three models to Adam, 0.001, ReLU, L1 and 50 respectively.

4.3 Experimental Results

4.3.1 Stocks Prediction

We have used past 10 years of Apple, Microsoft, IBM, Standard and Poor's data (2008-2018). The data consist of features like high, open, low, volume, close and status of all four companies. Status, which is a categorical variable (0 or 1) shows

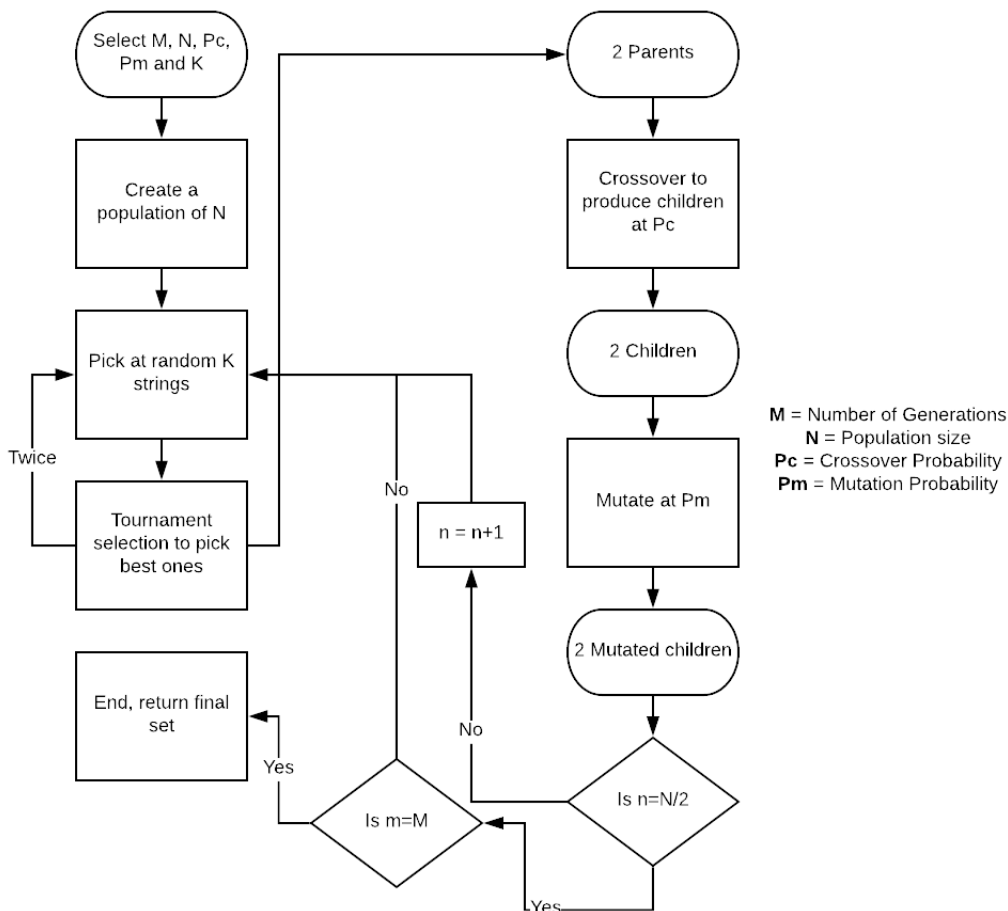


Figure 4.6: Genetic Algorithm Flowchart

whether today's closing price grew (1) or declined (0) from yesterday's closing price. We make use of sliding and non-sliding window technique to present the data. We made sure that there exists either strong positive or negative correlation between independent attributes like Apple (open, high, low, volume), Microsoft's, IBM's and S&P (open, high, low, volume, close) and Apple's closing price. The correlation heatmap is presented in figure 4.7.

The heatmap shows that there indeed exists a high positive or negative correlation between other variables and Apple's closing price.

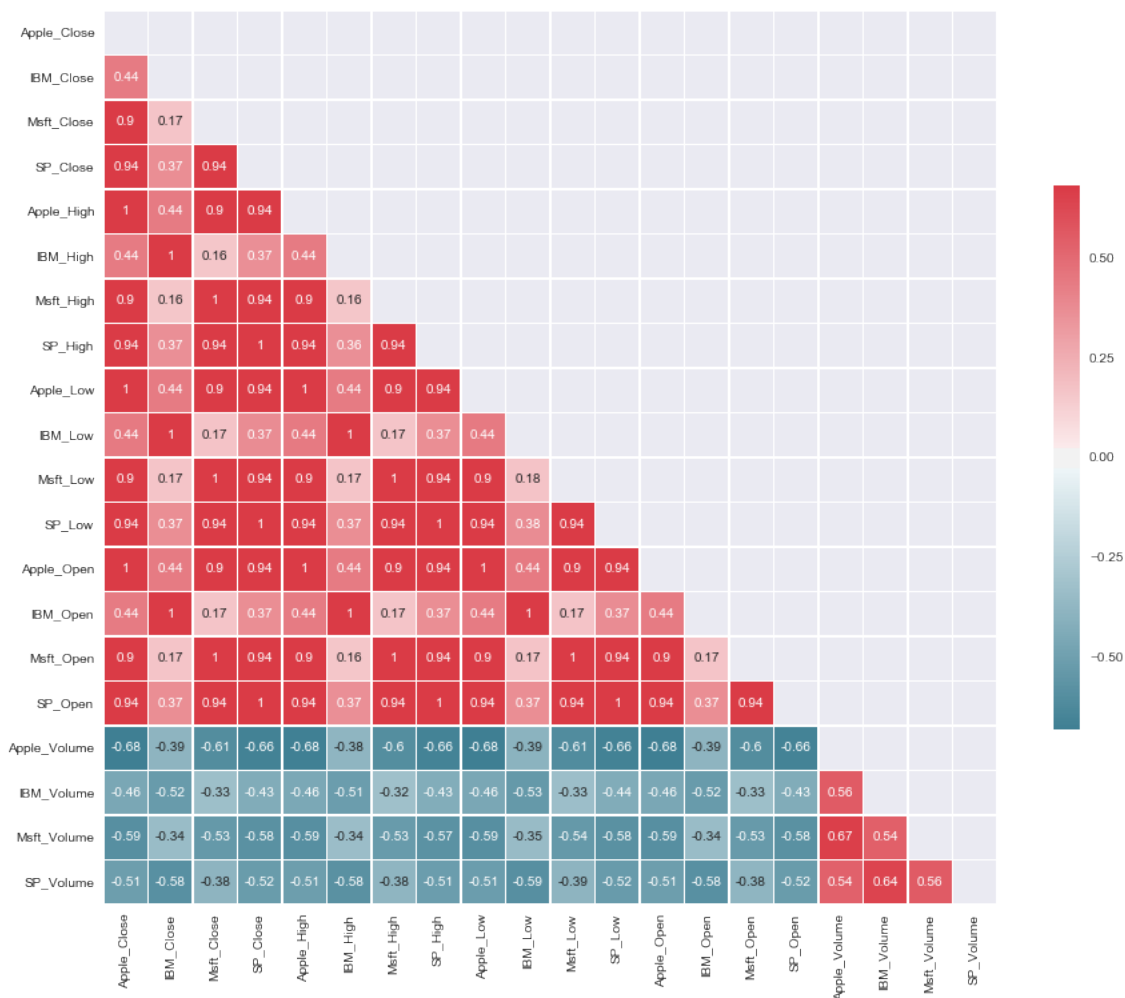


Figure 4.7: Attributes Correlation

4.3.1.1 Model Architecture

The hyper-parameters optimized models are fed with the non-sliding window (NSW) and sliding window (SW) data and we compare the performance of each case. Hyper-parameters of all the models are tuned using genetics algorithm and the optimized model summary is shown in table 4.1.

Non-Sliding Window (NSW)			
Models	Layers	Neurons	Weight Initialization
<i>NSW-ANN</i>	4	28, 14, 16, 1	LeCun Uniform
<i>NSW-RNN</i>	3	28, 14, 1	Normal
<i>NSW-LSTM</i>	3	28, 7, 1	Normal
Sliding Window (SW)			
Models	Layers	Neurons	Weight Initialization
<i>SW-ANN</i>	5	58, 6, 11, 13, 1	LeCun Uniform
<i>SW-RNN</i>	4	58, 12, 8, 1	Normal
<i>SW-LSTM</i>	3	58, 7, 1	Normal
Cost Function: Mean Squared Error			
Model Hyper-Parameters Tuning: Genetics Algorithm			
Model Optimization: Adam Optimizer			

Table 4.1: Different Models and their Characteristics

4.3.1.2 Results

Table 4.2 shows the mean squared error, r-squared and adjusted r-squared for three different models when fed with non-sliding window and sliding window data. From the table it is clear that LSTM has the least loss and highest r-squared and adjusted r-squared.

Non-Sliding Window (NSW)			
Model	MSE	R-Squared	Adjusted R-Squared
<i>NSW-ANN</i>	0.014	0.847	0.838
<i>NSW-RNN</i>	0.012	0.873	0.866
<i>NSW-LSTM</i>	0.009	0.881	0.874
Sliding Window (SW)			
Model	MSE	R-Squared	Adjusted R-Squared
<i>SW-ANN</i>	0.00150	0.962	0.96
<i>SW-RNN</i>	0.00110	0.978	0.975
<i>SW-LSTM</i>	0.00012	0.999	0.998

Table 4.2: MSE, R-Squared and Adjusted R-Squared

Non-Sliding Window (NSW)		
Model	APAE	Variance APAE
<i>NSW-ANN</i>	8%	36.24
<i>NSW-RNN</i>	7%	14.89
<i>NSW-LSTM</i>	6%	12.18
Sliding Window (SW)		
Model	APAE	Variance APAE
<i>SW-ANN</i>	2%	4.34
<i>SW-RNN</i>	1.85%	2.231
<i>SW-LSTM</i>	0.58%	0.278

Table 4.3: APAE and Variance of APAE

Table 4.3 shows the average prediction absolute error (APAE) and variance among the percentage error by different models. LSTM in both non-sliding and sliding window method performs best. In both the cases LSTM provides nearly 98% improvement in accuracy.

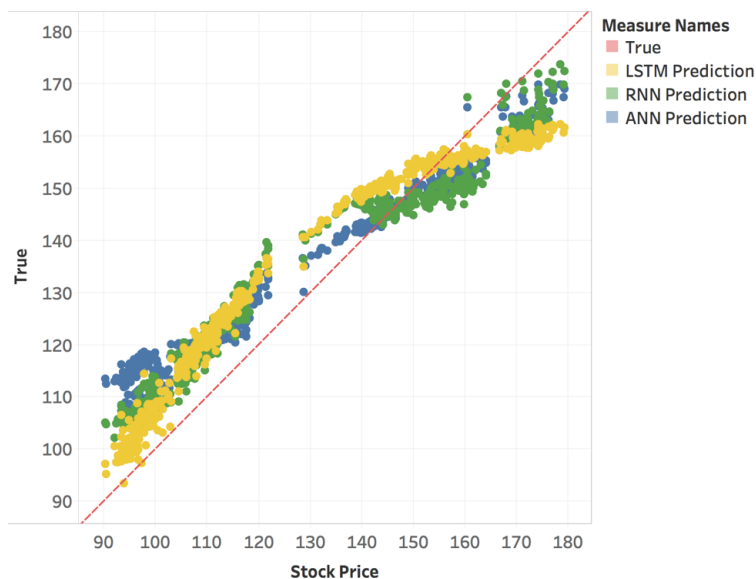


Figure 4.8: NSW Scatter Plot

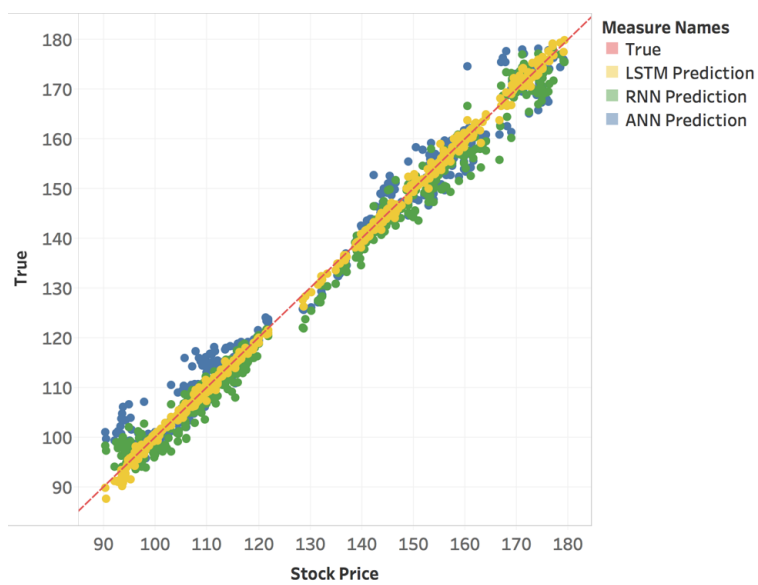


Figure 4.9: SW Scatter Plot

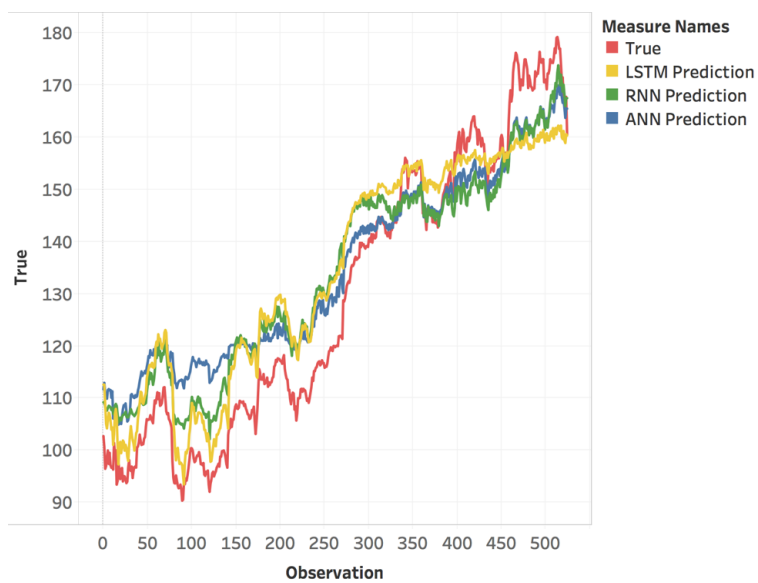


Figure 4.10: SW Prediction



Figure 4.11: SW Prediction

Figure 4.8 and 4.9 displays the scatter plot between the true values and the predicted values by the three models using non slided window features and slided window features respectively. One can see that the sliding window features play an essential role in predicting more accurate values than the non-sliding window features.

Figures 4.10 and 4.11 display the same results by overlapping the true and predicted values. The slided window features play an important role in accurate prediction of the test data. We can also see that hyper optimized LSTM model outperforms all the other models used for the purpose of prediction.

4.3.2 Currency Exchange

Prediction of how much a dollar will cost tomorrow can guide one's decision making and can be very important in minimizing risks and maximizing returns. Recurrent neural networks can be used in predicting the exchange rate between United States Dollar and Indian Rupees.

There are a lot of methods of forecasting exchange rates such as:

- **Purchasing Power Parity (PPP)**, which takes the inflation into account and calculates inflation differential.
- **Relative Economic Strength Approach**, which considers the economic growth of countries to predict the direction of exchange rates.
- **Econometric model** is another common technique used to forecast the exchange rates which is customizable according to the factors or attributes the forecaster thinks are important. There could be features like interest rate differential between two different countries, GDP growth rates, income growth rates, etc.
- **Time series model** is purely dependent on the idea that past behavior and price patterns can be used to predict future price behavior.

4.3.2.1 Time Series Prediction







 USD_INR.csv Date	 USD_INR.csv Price	 USD_INR.csv Open	 USD_INR.csv High	 USD_INR.csv Low	 USD_INR.csv Change %
8/10/2017	64.1650	63.8980	64.1750	63.8550	0.48000
8/9/2017	63.8600	63.7800	63.8600	63.7100	0.26000
8/8/2017	63.6920	63.7500	63.7850	63.6150	-0.23000
8/7/2017	63.8400	63.7100	63.8650	63.6480	0.26000
8/4/2017	63.6750	63.6700	63.7860	63.5720	-0.05000
1/4/1980	8.0500	8.0500	8.0500	8.0500	1.2600
1/3/1980	7.9500	7.9500	7.9500	7.9500	-0.6300
1/2/1980	8.0000	8.0000	8.0000	8.0000	0.0000

Figure 4.12: SW Prediction



Figure 4.13: SW Prediction

4.3.2.2 Time Series Prediction

The dataset used in this project is the exchange rate data between January 2, 1980 and August 10, 2017 shown in figure 4.12. The dataset displays the value of \$1 in rupees. We have a total of 13,730 records starting from January 2, 1980 to August 10, 2017. Over the period, the price to buy \$1 in rupees has been rising. One can see in figure 4.13 that there was a huge dip in the American economy during 2007–2008, which was hugely caused by the great recession during that period. It was a period of general economic decline observed in world markets during the late 2000s and early 2010s.

This period was not very good for the world's developed economies, particularly in North America and Europe (including Russia), which fell into a definitive recession. Many of the newer developed economies suffered far less impact, particularly China and India, whose economies grew substantially during this period.

4.3.2.3 Train-Test Split

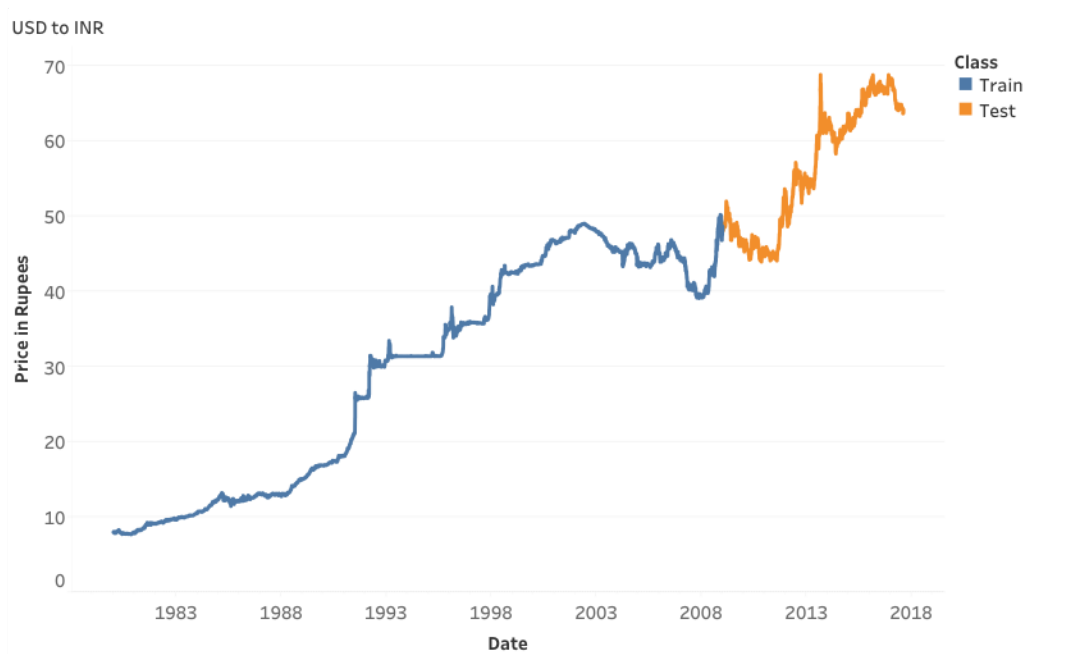


Figure 4.14: SW Prediction

For the purpose of training and validation we need to divide the dataset into training and test sets. It is very important to split train and test with respect to a certain date. In our experiment, we will define January 1, 2010, as our split date. The training data is the data between January 2, 1980 and December 31, 2009, which are about 11,000 training data points. The test dataset is between January 1, 2010 and August 10, 2017, which are about 2,700 points, shown in figure 4.14

In order to normalize the dataset ones need to fit and transform the training data and just transform the test data. The reason for which is not to assume that one knows the scale of the test data. Normalizing or transforming the data means that the new scale variables will be between zero and one.

4.3.2.4 Models

Sliding Window					
Models	Layers	Neurons	Weight Initializer	Window	
ANN	5	10, 7, 4, 3, 1	Lecun Uniform	7	
RNN	3	10, 14, 1	Lecun Uniform	7	
LSTM	3	10, 7, 1	Lecun Uniform	7	

Table 4.4: Hyper-Optimized Models for Currency Exchange Predictions

Table 4.4 shows the three models that we use for the purpose of prediction of currency exchange rates. All the three models are optimized using meta-heuristic technique called genetics algorithm. The optimized model configurations are listed in the table 4.4.

4.3.2.5 Results

Models	MSE	R-Squared	Adj R-Squared	APAE	Var APAE
ANN	$2.102e^{-3}$	0.937	0.921	3.14	3.27
RNN	$2.75e^{-4}$	0.977	0.963	0.428	0.762
LSTM	$4.5e^{-5}$	0.99	0.99	0.216	0.4275

Table 4.5: Model Performance Metrics for Currency Prediction

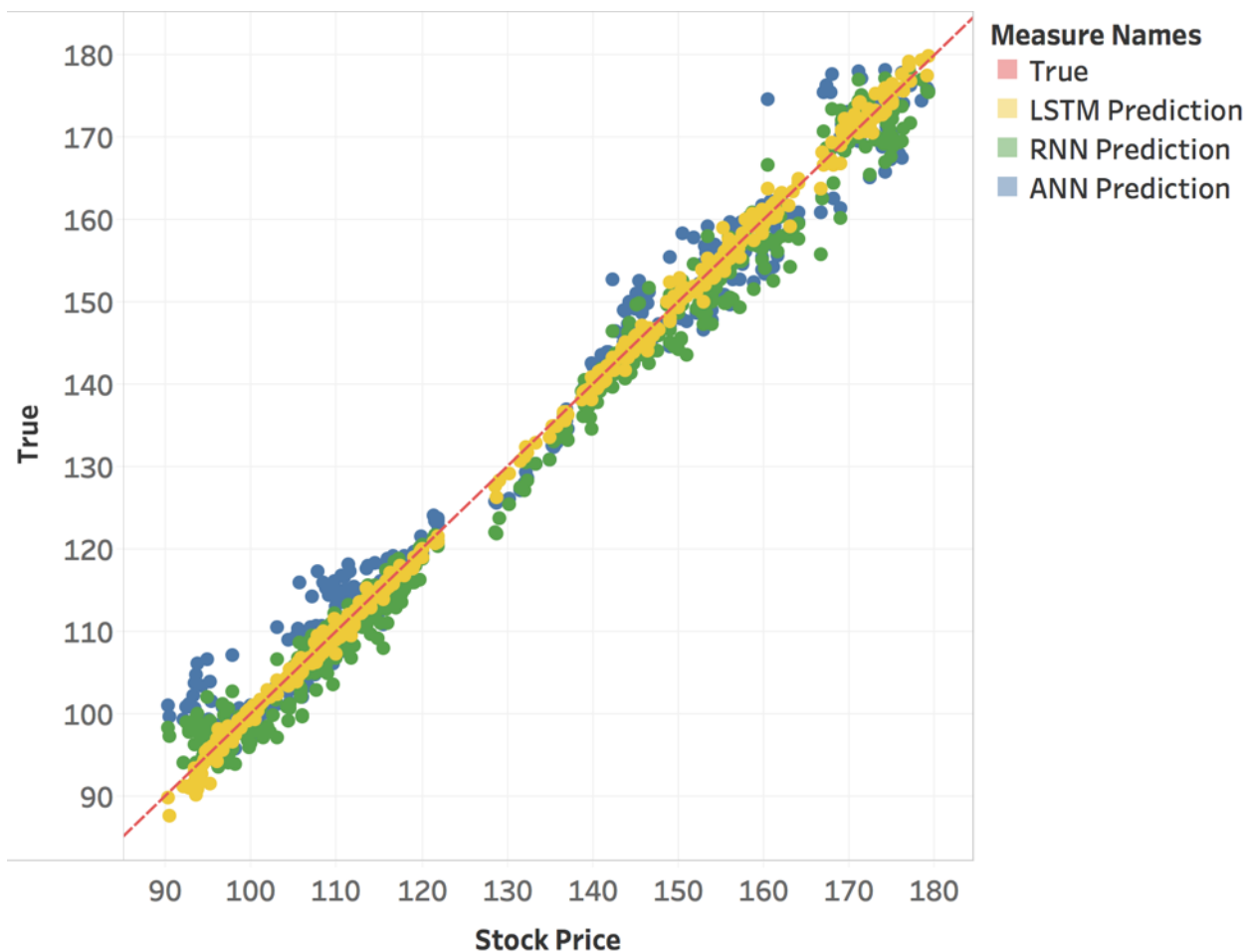


Figure 4.15: Model Predictions Scatter Plot

Table 4.5 shows the performance of three different hyperoptimized models used for predicting next day's currency exchange rate between USD and INR. The models used for the purpose of predictions were artificial neural network (ANN), recurrent neural network (RNN) and long short term memory neural network (LSTM).

The metrics used to measure model's performance are mean squared error (MSE), r-squared, adjusted r-squared, average prediction absolute error (APAE) and variance among the APAE for each model.

Table 4.5 clearly shows that LSTM has the least MSE, APAE and variance APAE compared to ANN and RNN. LSTM also has the highest r-squared and ad-

justed r-squared compared to ANN and RNN. This shows that LSTM is able to predict future values more accurately than the rest of the models. Such accuracy is due to LSTM's capability of learning long term dependencies.

The predicted results by all the three models are shown in figure 4.15. The diagonal red line displays the ground truth and the circles represents the predicted values. Blue, green and yellow represent ANN, RNN and LSTM's predictions respectively. Accuracy is judged by the closeness of the circles to the diagonal red line. It is clear in the figure that LSTM predictions are much closer to the diagonal line than compared with the other two models.

4.3.3 Location Prediction

In this section we will discuss the methodology used to predict a user's future locations. The dataset used in this project is the same, which was used in chapter 3. The data was collected from Google Maps¹ application installed on a mobile phone for a period of 6 months.

The user was a student at the University of Texas at Arlington, while the data was collected. A smartphone, with google maps installed and location history set to "On", we collected GPS data from the user for a period of six months (February 2017 - July 2017). During those six months the user mainly browsed various locations around Arlington, Texas. At the end of the sixth month the database collected *9,185 GPS observations*. Each of the tuple recorded was of the form,

< Latitude, Longitude, Hour, Day of week, Month of year >

¹<https://www.google.com/maps>

with a precision of 7 decimal points for Latitude and Longitude as shown in Figure 3.8, this greater precision in GPS locations helps in achieving greater precision in location prediction. We also appended weather information to the GPS dataset. We have obtained the weather information on the users location through NOAA(National Oceanic and Atmospheric Administration)²website.

The objective is to classify the locations using ANN and LSTM. We also use Support Vector Machine (SVM), K-Nearest Neighbors and Random Forest classifiers to set up benchmark values.

4.3.3.1 Location Clustering

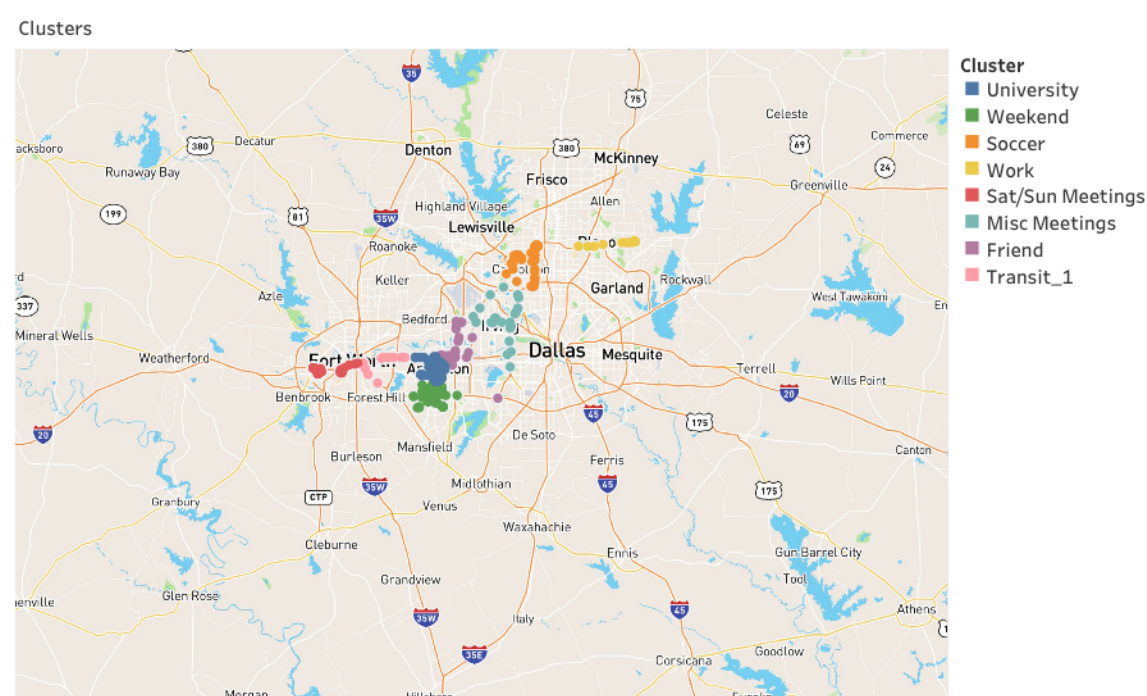


Figure 4.16: User's Clusters

²<https://www.ncdc.noaa.gov/cdo-web/datasets/LCD/stations/WBAN:53907/detail>

We used varied K-Means algorithm from chapter 2 to cluster user's locations. We found 8 and 10 as optimal number of clusters in the dataset. The clusters are shown in figure 4.16. We manually labelled the clusters after inspection such that it is easier to differentiate between them.

4.3.3.2 Results

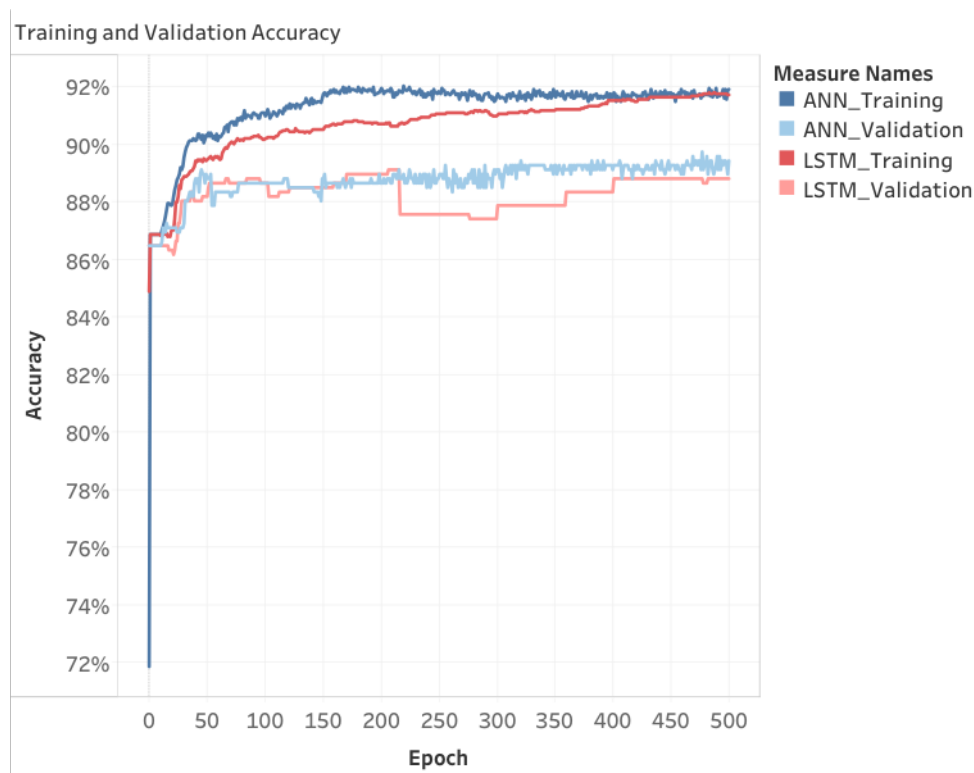


Figure 4.17: Training and Validation Accuracy

Figure 4.17 and 4.18 display the training/ validation accuracy and loss. Looking at the figure 4.17 one can see that ANN is performing better than LSTM. The validation accuracy by ANN is 91% as compared to 90% by LSTM. Loss, which is shown in figure 4.18 by ANN is lower than LSTM model.

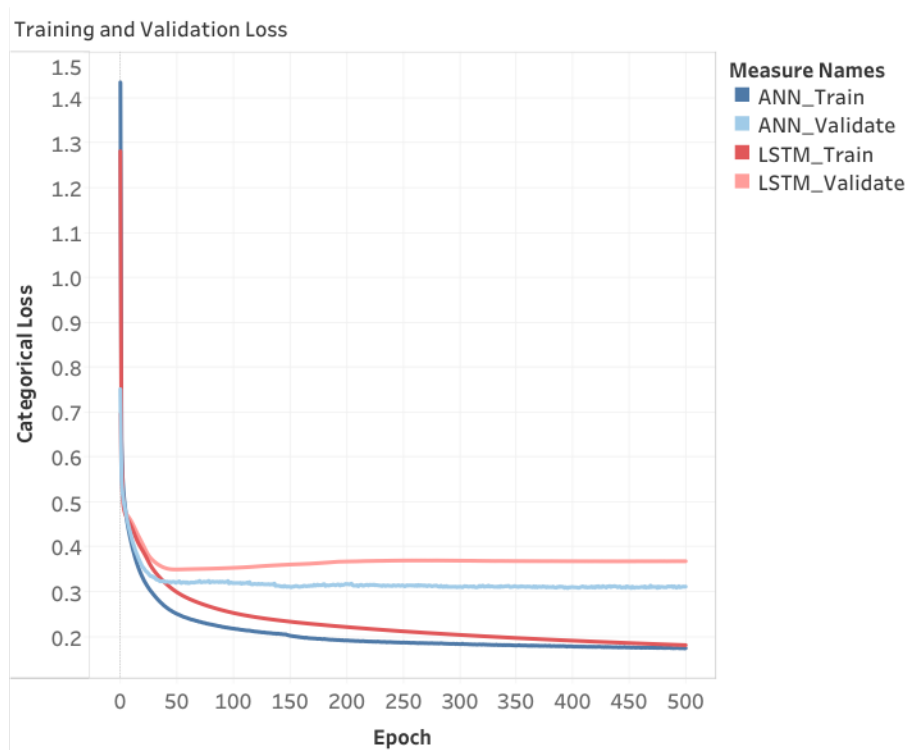


Figure 4.18: Training and Validation Loss

The validation data used in this experiment was 20% of the training dataset. The testing dataset was never shown to the models and the accuracy on this dataset by each model is shown in figure 4.19. Optimized LSTM performs the best out of all the models by predicting 90% correct location clusters. Optimized ANN is able to classify 88% correct locations and traditional machine learning models like SVM, K-NN, Random Forest classifier predicts about 82, 78 and 74% correct locations respectively.

Figure 4.20 shows time taken by each model to train. LSTM takes about 500 seconds to train for 500 epochs and is almost 5 times slower than ANN, which takes about 100 seconds to train.

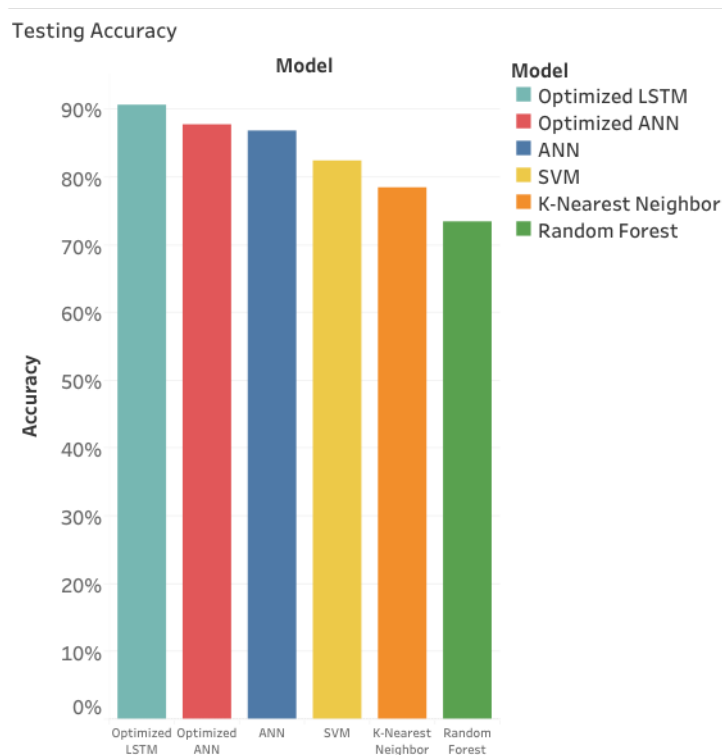


Figure 4.19: Testing Data Accuracy

4.4 Conclusion

In this chapter we discussed about feed forwards neural networks, recurrent neural networks and a special type of RNN known as long-short term neural network. We also discussed about a meta-heuristic optimization algorithm called genetics algorithm. We used the three typed of neural networks on three different domains to predict future time series values. The domains we considered for the experiments were stocks, currency and GPS locations.

We optimized all the models using genetics algorithm and we witnessed that LSTM outperformed regular neural networks and traditonal machine learning methods. It can be difficult to train regular feed forward neural networks and RNNs to solve problems that require learning long-term temporal dependencies. This is because the gradient of the loss function decays exponentially with time (called the

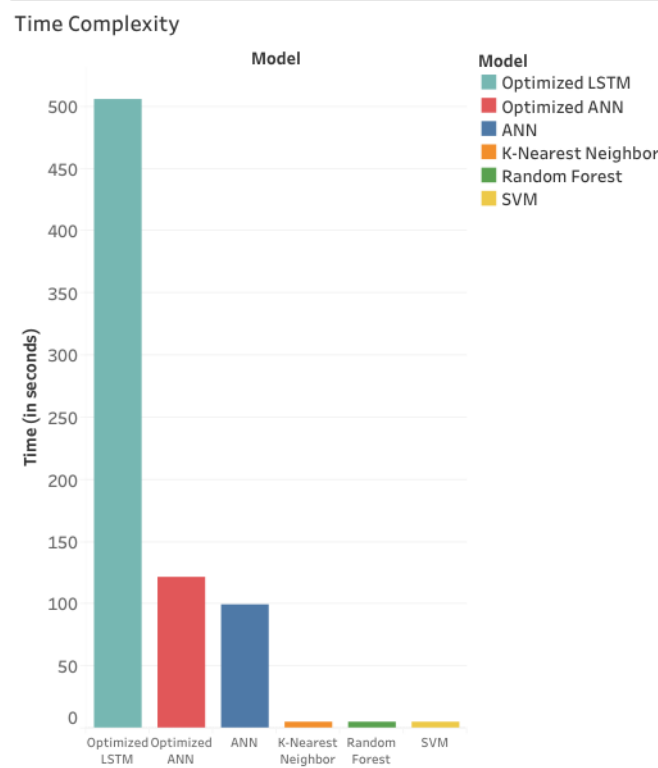


Figure 4.20: Time Taken to Train

vanishing gradient problem). As discussed in section 4.2.4, LSTMs use special units in addition to standard units, which lets them learn longer-term dependencies.

CHAPTER 5

CONCLUSIONS

5.1 Summary of Contributions

This dissertation was focused on prediction of Spatio-Temporal data like GPS locations, temporal data like stock prices and currency exchange rates. We tackled the problems of prediction by implementing state-of-the-art models like Feed Forward Neural Networks, Recurrent Neural Networks and Long Short Term Neural Networks. Each of the model's hyperparameters were optimized using meta-heuristic optimization technique called Genetics Algorithm.

In Chapter 2, we made use of varied K-means clustering algorithm and Hidden Markov Model to predict a user's future locations. This work describes correctly classifying future locations of a user such that we can answer questions like "*Where is the user most likely to be when it is a Monday?*" or day and time-specific queries like "*Where is the user most likely to be between 6:00 pm and 9:00 pm on Saturdays?*". The first step is to cluster locations and to solve this we use varied K-means algorithm, which unlike traditional K-means, sophisticatedly decides that number of significant clusters of a user. These clusters are then considered as visible states in the HMM and the hidden states are the days of the week and time of the day.

In Chapter 3, we made use of Feed Forward Neural Networks to predict the future locations of the user. In this work we concentrated on both regression and classification of user's locations.

In Chapter 4, we shifted our interest from just spatio-temporal data prediction to purely temporal data predictions. The domain of interest in this chapter was

stock prices (Apple Inc.) and currency exchange prices (between USD and INR). We make use of Feed Forward Neural Networks, Recurrent Neural Networks and Long Short Term Neural Networks. The hyper parameters of each model is optimized using Genetics Algorithm and we also benchmark the results using traditional machine learning techniques like Support Vector Machines, K-Nearest Neighbors and Decision Trees.

REFERENCES

- [1] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi, “Human mobility, social ties, and link prediction,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. Acm, 2011, pp. 1100–1108.
- [2] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, “Understanding mobility based on gps data,” in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 312–321.
- [3] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, “Mining interesting locations and travel sequences from gps trajectories,” in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 791–800.
- [4] Y. Zheng, X. Xie, and W.-Y. Ma, “Geolife: A collaborative social networking service among user, location and trajectory.” *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, 2010.
- [5] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.
- [6] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.
- [7] D. J. Montana and L. Davis, “Training feedforward neural networks using genetic algorithms.” in *IJCAI*, vol. 89, 1989, pp. 762–767.
- [8] L. Bianchi, L. M. Gambardella, and M. Dorigo, “An ant colony optimization approach to the probabilistic traveling salesman problem,” in *International Conference on Parallel Problem Solving from Nature*. Springer, 2002, pp. 883–892.

- [9] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM computing surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.
- [10] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.
- [11] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.
- [12] L. E. Peterson, “K-nearest neighbor,” *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.
- [13] L. Di Persio and O. Honchar, “Artificial neural networks architectures for stock price prediction: comparisons and applications,” *International Journal of Circuits, Systems and Signal Processing*, vol. 10, pp. 403–413, 2016.
- [14] D. Ashbrook and T. Starner, “Learning significant locations and predicting user movement with gps,” in *Wearable Computers, 2002. (ISWC 2002). Proceedings. Sixth International Symposium on*. IEEE, 2002, pp. 101–108.
- [15] —, “Using gps to learn significant locations and predict movement across multiple users,” *Personal and Ubiquitous computing*, vol. 7, no. 5, pp. 275–286, 2003.
- [16] W. Mathew, R. Raposo, and B. Martins, “Predicting future locations with hidden markov models,” in *Proceedings of the 2012 ACM conference on ubiquitous computing*. ACM, 2012, pp. 911–918.
- [17] J. Yang, J. Xu, M. Xu, N. Zheng, and Y. Chen, “Predicting next location using a variable order markov model,” in *Proceedings of the 5th ACM SIGSPATIAL International Workshop on GeoStreaming*. ACM, 2014, pp. 37–42.
- [18] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

- [19] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.
- [20] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, “Destination prediction by sub-trajectory synthesis and privacy protection against such prediction,” in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 254–265.
- [21] M. Morzy, “Mining frequent trajectories of moving objects for location prediction,” in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2007, pp. 667–680.
- [22] G. Van Brummelen, *Heavenly mathematics: The forgotten art of spherical trigonometry*. Princeton University Press, 2012.
- [23] H. H. Bui, “A general model for online probabilistic plan recognition,” in *IJCAI*, vol. 3. Citeseer, 2003, pp. 1309–1315.
- [24] S. R. Eddy, “Profile hidden markov models.” *Bioinformatics (Oxford, England)*, vol. 14, no. 9, pp. 755–763, 1998.
- [25] G. Yavaş, D. Katsaros, Ö. Ulusoy, and Y. Manolopoulos, “A data mining approach for location prediction in mobile environments,” *Data & Knowledge Engineering*, vol. 54, no. 2, pp. 121–146, 2005.
- [26] R. Agrawal and R. Srikant, “Mining sequential patterns,” in *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*. IEEE, 1995, pp. 3–14.
- [27] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, “Wherenext: a location predictor on trajectory pattern mining,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 637–646.

- [28] N. Pant and R. Elmasri, “Detecting meaningful places and predicting locations using varied k-means and hidden markov model.”
- [29] P. Bahl and V. N. Padmanabhan, “Radar: An in-building rf-based user location and tracking system,” in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. Ieee, 2000, pp. 775–784.
- [30] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, “A probabilistic approach to wlan user location estimation,” *International Journal of Wireless Information Networks*, vol. 9, no. 3, pp. 155–164, 2002.
- [31] P. Bahl, V. N. Padmanabhan, and A. Balachandran, “Enhancements to the radar user location and tracking system,” *Microsoft Research*, vol. 2, no. MSR-TR-2000-12, pp. 775–784, 2000.
- [32] E. Cho, S. A. Myers, and J. Leskovec, “Friendship and mobility: user movement in location-based social networks,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1082–1090.
- [33] F. Shih and J. Boortz, “Understanding people’s preferences for disclosing contextual information to smartphone apps,” in *International Conference on Human Aspects of Information Security, Privacy, and Trust*. Springer, 2013, pp. 186–196.
- [34] Z. K. Gurmu and W. D. Fan, “Artificial neural network travel time prediction model for buses using only gps data,” *Journal of Public Transportation*, vol. 17, no. 2, p. 3, 2014.
- [35] Q. Liu, S. Wu, L. Wang, and T. Tan, “Predicting the next location: A recurrent model with spatial and temporal contexts.” in *AAAI*, 2016, pp. 194–200.

- [36] T. S. dos Santos, D. Mendes, and R. R. Torres, “Artificial neural networks and multiple linear regression model using principal components to estimate rainfall over south america,” *Nonlinear Processes in Geophysics*, vol. 23, no. 1, p. 13, 2016.
- [37] B. Widrow and E. Walach, *Adaptive inverse control, reissue edition: a signal processing approach*. John Wiley & Sons, 2008.
- [38] H. Bilen and A. Vedaldi, “Weakly supervised deep detection networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2846–2854.
- [39] L. A. Saaf, “Feedforward neural networks for image classification,” 1992.
- [40] Y. Huang, Z. Wu, R. Li, H. Meng, and L. Cai, “Multi-task learning for prosodic structure generation using blstm rnn with structured output layer,” *Proc. Interspeech 2017*, pp. 779–783, 2017.
- [41] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.
- [42] A. J. Yepes, J. Tang, and B. S. Mashford, “Improving classification accuracy of feedforward neural networks for spiking neuromorphic chips,” *arXiv preprint arXiv:1705.07755*, 2017.
- [43] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [44] X. Li, H. Xie, L. Chen, J. Wang, and X. Deng, “News impact on stock price return via sentiment analysis,” *Knowledge-Based Systems*, vol. 69, pp. 14–23, 2014.
- [45] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, “Predicting stock market index using fusion of machine learning techniques,” *Expert Systems with Applications*, vol. 42, no. 4, pp. 2162–2172, 2015.

- [46] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp, “Evaluating multiple classifiers for stock price direction prediction,” *Expert Systems with Applications*, vol. 42, no. 20, pp. 7046–7056, 2015.
- [47] J. Zahedi and M. M. Rounaghi, “Application of artificial neural network models and principal component analysis method in predicting stock prices on tehran stock exchange,” *Physica A: Statistical Mechanics and its Applications*, vol. 438, pp. 178–187, 2015.
- [48] S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon, and K. Soman, “Stock price prediction using lstm, rnn and cnn-sliding window model,” in *Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on*. IEEE, 2017, pp. 1643–1647.
- [49] K. Chen, Y. Zhou, and F. Dai, “A lstm-based method for stock returns prediction: A case study of china stock market,” in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2823–2824.
- [50] A. M. Rather, A. Agarwal, and V. Sastry, “Recurrent neural network and a hybrid model for prediction of stock returns,” *Expert Systems with Applications*, vol. 42, no. 6, pp. 3234–3241, 2015.
- [51] E. Guresen, G. Kayakutlu, and T. U. Daim, “Using artificial neural network models in stock market index prediction,” *Expert Systems with Applications*, vol. 38, no. 8, pp. 10 389–10 397, 2011.
- [52] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyperparameter optimization,” in *Advances in neural information processing systems*, 2011, pp. 2546–2554.
- [53] W. Fenghua, X. Jihong, H. Zhifang, and G. Xu, “Stock price prediction based on ssa and svm,” *Procedia Computer Science*, vol. 31, pp. 625–631, 2014.

- [54] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [55] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

BIOGRAPHICAL STATEMENT

Neelabh Pant was born in Nainital, India in 1993. He received his Bachelor in Computer Science and Engineering from Birla Institute of Applied Sciences, India, in 2013. In 2014, he started his Masters studies at the University of Texas at Arlington. His Master's Thesis titled *Performance Comparison of Spatial Indexing Structures For Different Query Types* concentrated on the comparison of performance of spatial indexing structures like R-Trees and its variants. In 2015, he started his studies to pursue his Ph.D at the University of Texas at Arlington. His current research interest include, Artificial Intelligence, Deep Learning, Computer Vision, Machine Learning, and Natural Language Processing.