FORECASTING USER BEHAVIORS IN CALL DETAIL RECORDS USING LSTM MODELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HASAN KOCAMAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

FEBRUARY 2021

ABSTRACT

FORECASTING USER BEHAVIORS IN CALL DETAIL RECORDS USING LSTM MODELS

Kocaman, Hasan
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. İsmail Hakkı Toroslu

February 2021, 69 pages

City planners, governors and mobile phone operators utilize quite from Call Detail Records (CDR) data in the fields like optimizations of traffic congestion, event detection, billing and advertisement policies. Both individual and crowd analyses help improving the quality of services.

In this thesis, we present regression and classification analysis for three main problems. In regression analysis, we experiment on call counts and call time-sums of specified numbers of users. We propose cluster based and outlier separating models in these two tasks for the purpose of improving the results of individual user-based models comprised of various Long Short Term Memory(LSTM) layers. In the classification analysis, on the other hand, we present models that predict next locations on the trajectories of the users. We improve the results of base LSTM model with two-predictions-at-once approach. The analyses show that recurrent neural networks work well with sequential data and optimizations on top of the models yield promising results.

Keywords: Call Detail Records, Artificial Neural Networks, Cluster Analysis, Sequential Data

ÖZ

ARAMA AYRINTILARI KAYITLARINDA KULLANICI DAVRANIŞLARININ LSTM MODELLERİ İLE TAHMİN EDİLMESİ

Kocaman, Hasan
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. İsmail Hakkı Toroslu

Şubat 2021 , 69 sayfa

Şehir planlamacıları, yöneticileri ve mobil telefon operatörleri Arama Ayrıntıları Kayıtları (CDR) verilerinden trafik tıkanıklığı iyileştirmeleri, olay belirleme, faturalandırma ve reklam politikaları gibi alanlarda oldukça yararlanmaktadır. Bireysel ve topluluk analizleri hizmet kalitesini artırmaya yardımcı olmaktadır.

Bu tezde, üç ana problem için regresyon ve sınıflandırma analizi sunmaktayız. Regresyon analizinde, belirli sayıda kullanıcıların arama sayıları ve toplam-zamanları üzerine deneyler yapmaktayız. Bu iki görevde, çeşitli uzun kısa zamanlı bellek (LSTM) katmanlarından oluşan bireysel kullanıcı bazlı modellerin sonuçlarını iyileştirmek amacıyla kümeleme tabanlı ve aykırı gözlemleri ayrıştıran modeller öne sürmekteyiz. Diğer yandan, sınıflandırma analizinde ise kullanıcıların gezingelerindeki bir sonraki konumlarını tahmin eden modeller ileri sürmekteyiz. Temel LSTM modelinin sonuçlarını tek-seferde-iki-tahmin yaklaşımıyla iyileştirmekteyiz. Analizler, tekrarlayan sinir ağlarının sıralı verilerde iyi çalıştığını ve modellerin üzerinde yapılan iyileştirmelerin, umut vaad eden sonuçlar verdiğini göstermektedir.

Anahtar Kelimeler: Arama Ayrıntıları Kayıtları, Yapay Sinir Ağları, Kümeleme Analizi, Sıralı Veri

To my beloved nieces, Göksenin and Beren

ACKNOWLEDGMENTS

I would like to present my sincere gratitude to Prof.Dr. Ismail Hakkı Toroslu for accepting me as his student, sincere friendly attitudes towards me and most importantly mentoring throughout this work. Without his guidance, I would not find the ways to overcome the troubles and finish this thesis.

I would like to express my thanks to Prof. Dr. Pınar Karagöz and Assist. Prof. Dr. Engin Demir for accepting to be my jury members and for their valuable comments and suggestions.

I would like to thank my friends and colleagues, Can, Umut, Ömür and Mehmet for encouraging me to finish this work. Without their motivating words, this thesis could not be finished, too.

I would also like to thank my friends in this department, Aybars, Alara, Onur and Ersel for their sincere friendship and accompany throughout the courses. Their helpfulness made everything much easier.

Last but not the least, I wish to express my hearty gratitude to my beloved family for being by my side in every decision I have taken up to this moment and for their endless support during this work.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	
ANN	Artificial Neural Networks
ARIMA	AutoRegressive Integrated Moving Average
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
RNN	Recurrent Neural Networks
DTW	Dynamic Time Warping
DBSCAN	Density-based Spatial Clustering of Applications with Noise
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
ADF Test	Augmented Dickey-Fuller Test

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

After the huge increase of mobile phone usage starting especially from 2000s, mobile phones have become an indispensable part of daily life. It is important not only for the users, but also for telecommunication companies because of the data generated by the usage of mobile phones. The analyses produced by telecommunication data can be beneficial in many research areas like fraud detection, advertisement policy enhancement, demographic analysis etc.

As one of the valuable data of telecom companies, Call Detail Records (CDR) include detailed information of caller and callee (ids, time of call, type of transaction, location etc.). While these data are being used by telecom companies in the processes such as billing, precise marketing or customer experience, CDR may also be used in areas like political campaigns or social behavior analysis outside telecom companies [8].

Outgoing calls of the users in CDR data may produce signs for the behavioral analysis for individuals and crowds. Analyses on number of calls or total time of callings that users have in one day or in a period is helpful for understanding the usage of mobile phones. This analysis is then utilized in telecom companies for detecting good or bad customers in terms of usage and loyalty to the company for billing policies. Companies also may detect big events like earthquake or explosions and change their service policy with the prior signs in the increase of calls.

Another analysis can be done based on the locations of base stations and connected users to them. The patterns of movements in a period help analyzing the behaviors of

users or society. With the individual analyses, company may present good advertisement policies according to frequent places that users visit. Besides, detecting crowds with the help of base stations is critical for city governors to make good decisions for traffic congestion, city planning etc. While analysis on base stations is one tool for increasing the quality of service of the telecom companies, it may also give help to the city governors.

We have analyzed raw CDR data gathered from one of the well-known telecom companies in Turkey. Our work includes specific deep learning methodologies and clustering techniques for two set of the problems, namely regression analysis. In these sets, we analyze **number of calls** and **total time of calls** benefiting from recurrent neural networks. In the other set of the problems, we analyze the **locations of the users**. We try to beat the benchmark model of frequent place selection with deep learning approaches. Thus, we try to figure out how successful and promising the modern and improving deep learning technologies is for the analyses on Big Data, CDR specifically.

1.2 Proposed Methods and Contributions

Our analyses for CDR data are comprised of two main sections:

- In the regression analysis, we predict one time interval ahead of call counts and call time-sum of user groups. We both use raw (non-discretized) data and discretized data (dividing the day to 4-part time interval) for 30-day of 100, 1000 and 10000 users, respectively.
- In the classification analysis, on the other hand, we predict the next locations of individuals in each user group. Trajectories of the users are trimmed according to average call numbers of total users in each group.

Our initial studies for the regression analysis start with comparisons of conventional statistical methods for time-series to sequential data analysis methods using recurrent neural networks, especially LSTM models. First analyses are related to the models inspecting the **individual patterns.** Every user is trained with her/his own data and

this training is not sufficient for promising results. As a contribution to the literature, we first define **cluster based LSTM models** and change the approach of handling training data. We increase the amount of training data for each user having same kind of behaviors within clusters. After this clustering process, we see that improvements are not as we expect. For the purpose of being more accurate in the joining of data, we detect **noises or outliers** in user groups and separate them from the training phase. To the best of our knowledge, this forms the second contribution for CDR data analysis in terms of user behaviors.

In the classification analysis, we experiment on finding the next locations on the trajectories of the users. The benchmark model finds the most frequent places in the movements of the users. We compare this approach to well-known deep learning models, MLP and LSTM. Although deep learning approaches seem suitable for making predictions in the next location problem, we are not content with the results. We propose a **two-predictions-at-once** models on top of LSTM model. As a first contributing model, we add the option of second probable location to the predictions. Another option as a model is making the current location join to the predictions. The models first find the predicted location according to the base model and add current or second probable location to the predictions. This forms the third contribution to CDR data analysis in terms of next location prediction.

1.3 The Outline of the Thesis

The organization of the thesis is as follows:

In Chapter 2, we present detailed literature survey related with proposed methods in this work regarding time series analysis and mobile data predictions with recurrent neural networks.

In Chapter 3, we explain well-known deep learning approaches used in analyses and clustering methods that help a lot for improving the results. We also present data set representation.

Chapter 4 gives the problems defined for CDR data analysis and presents the types of

models for solving the problem types. We also give details on the error metrics for measuring the accuracy of the models.

In Chapter 5, we present the experimental results and discussions of these results.

Chapter 6 concludes the thesis and gives recommendations for future directions.

CHAPTER 2

RELATED WORK

2.1 Time Series Analysis with Statistical Methods and Deep Learning Approaches

In a long dated work [9], when RNNs were not popular for time series analysis, S.L. Ho et al. compared ARIMA model with ANN and RNN in failure analysis of a repairable system. In their work, feed forward model performed poorly in both short term forecasting and error rates. However, ARIMA model and RNN model yielded promising results for being used in time series analysis of failures in the repairable system. With this analysis, they suggested that prediction of time of failures could be beneficial in maintenance processes.

Using an ARIMA-Based Recurrent Network system [10], J-H. Wang and J-Y. Leu analyzed price trend in Taiwan stock market for forecasting 6-week ahead of the data. They utilized ACF plots for making data stationary and defined the structure of the data as ARIMA(1,2,1). They fed second-differenced data and residuals of this data as input to recurrent network. The results showed that they predicted stock prices with acceptable accuracy.

Ç.H.Aladağ et al. constructed a hybrid model [11] which combined ARIMA and Elman's Recurrent Neural Network (ERNN) [12] for the analysis of Canada Lynx data set. The linear part of data set was analyzed with Box-Jenkins Models and the residuals of linear part was fitted to ERNN. They compared the method with two other hybrid models and also FNN. RMSEs of the models showed the superiority of the hybrid method in forecasted values.

In [13], I.Khandelwal et al. analyzed four different data sets with their hybrid method

which utilized Discrete Wavelet Transform (DWT) for separating linear and nonlinear parts as high and low frequency components. They used ARIMA and ANN like Zhang's proposed method [14] in order to combine linear components and residuals as nonlinear. Their DWT based hybrid model outperformed ARIMA, ANN and Zhang's hybrid method in terms of MAE and MAPE in all of the data sets.

In [15], S-L. Yu and Z. Li compared ARIMA, RNN and hybrid ARIMA-RNN models on stock prices of five indices. Their hybrid model was composed of moving average filter which separated low volatility part as linear time series and the other part as non-linear time series. They applied ARIMA model to the linear part and for the non-linear part, RNN model was used. They concluded that, in all of the metrics they evaluated, hybrid ARIMA-RNN model outperformed the other base models while RNN was better than ARIMA.

M.Rhanoui et al. analyzed financial time series data with a subgroup of ARIMA models, namely Random Walk and compared its performance with LSTM based deep network [16]. They described why they utilized these models in predictions of budget data set and how they fine-tuned the parameters. Their results showed that deep learning methods were suitable for analysis of financial time series because of nonlinear nature of the data set.

In their work [17], M.Ashour and R.Helmi compared the prediction performance of ERNN and ARIMA(1,0,0) model in U.A.E. oil production data set. Error scores of ERNN showed that the model outperformed the traditional ARIMA model. They concluded that learning and self-adaptation nature of neural networks was clearly the reason of this superiority.

Like as it was applied in [13], R.Madan and P.S.Mangipudi applied DWT on computer network traffic data set for decomposing linear and nonlinear parts [18]. They used ARIMA on linear part and RNN on nonlinear part for forecasting daily, hourly and 5-minute time series. NRMSE scores of the method indicated that it outperformed base models and yielded promising results for using it in data centers.

A.S.Temur et al. used Turkey housing prices data to determine future estimates of housing sales [19]. They compared ARIMA model with different parameters, LSTM

architecture with different epochs and the hybrid model of these two models. Considering MAPE values of these three models, they concluded that the hybrid model outperformed base models. This study was first in Turkey on the field of housing prices, which yielded accurate results.

In their study [20], Namini et al. used financial data sets to compare the performance of ARIMA and LSTM in multi-step forecasting. Besides, they examined the effect of epoch number to the success rates of predictions. Their results indicated that LSTM was superior to ARIMA model with a high margin and epoch numbers had no crucial effect in predictions for financial data.

In detailed study of them [21], H.Hewamalage et al. presented current status of seasonal time series analysis. They indicated that beyond traditional methods like ARIMA and Error-Trend-Seasonality (ETS), RNNs have taken their places in the literature with different RNN units (ERNN, GRU or LSTM) applied in different problems. They also mentioned the importance of attention mechanism that can be used in seasonal data and ensemble methods for global solutions to the problems. Their methods for different competition data sets were composed of variations of sequence to sequence models, stacked models and also seasonal and trend decomposition model. They examined the effect of different types of hyper-parameters and optimizers to reach the optimal forecast results and concluded that RNNs yielded performances that was over benchmark results.

2.2 Prediction on Mobile Data with Recurrent Neural Networks

In [22], L.Yeng et al. constructed a travel management recommendation system both for the travelers to visit where and when, and for the authorities to decide city and traffic planning. In the location recommendation part of their optimization, they utilized CDR data for next-location prediction using RNN architecture.

In [23], N.Chen et al. investigated the prediction of tourists' next visit locations in the region of Andorra using many deep learning methods. The purpose of the work was to predict the fifth location of users which had tourist nationality. After many feature engineering methods were applied to the raw data set, the best classification

algorithm was RNN with the success rate of 94.8%. The reason for this success rate was the RNN's capability of handling sequential and different-size input.

M.S.Mahdizadeh and B.Bahrak analyzed CDR data of 12 users in Tehran for the problem of next location prediction [24]. They compared two RNN architectures, classification and regression RNN, to the most frequent next location prediction and Markov Decision Chain. The window size for input sequence changed according to four methods in data preparation phase; 1, fixed size, dynamic and hybrid method. The regression RNN framework outperformed the traditional models by 74% to 55% improvement in error rates.

DeepMove, which was an attention based recurrent neural network, was proposed by J. Feng, Y. Li et al. [25] for solving the heterogeneity and sparsity problems in three mobile phone data sets. One of the data sets was CDR gathered from China. Their model was a combination of attention mechanism with GRU framework. While GRU units capture the meanings in current trajectory, historical attention module chooses the best relevant historical records. This framework performed better than baseline methods.

In his thesis [26], Y.Leng analyzed different CDRs for three purposes. In the first part of the work, as a mobility pattern mining study, mobility prediction of users was investigated for inferring workplaces and home places. In the second part, using recurrent neural networks, next location prediction of users was proposed as a work of mobility prediction. The method for generating user trajectories was to use cell tower traces as sentences. These sentences were fed into recurrent neural network for prediction next cell tower. In the last part of his study, for the purpose of changing behaviours of users, an optimization method regarding both user preferences and other constraints like traffic congestion was proposed as a proof-of-concept model.

In [27], A.Crivellari and E.Beinat explored the next location prediction of short-term visitors in the region of Italy using CDR data via comparing LSTM-based model with three types of Markov model. Their LSTM model was composed of preprocessing step in which trajectories were turned into inputs for neural network, embedding step of these sequences into dense vectors and 2-layered LSTM network followed by a softmax layer in which next location prediction was analyzed. Different types of

comparisons showed that the deep learning model outperformed with respect to all of the three conventional Markov Models and error distance was also meaningful when compared to baseline models. The work is important for smart city planners and tourists trying to make holiday plans.

In [28], H.D. Trinh et al. analyzed mobile traffic data using LSTM based architecture for multi-step prediction of usage of cellular network. They applied only data aggregation to their raw data collected from the users connected to base stations so that the speed of their prediction were high compared to other preprocessed data. Their LSTM method provided superior accuracy to ARIMA and Feed Forward Neural Network model. With their methodology, the process of traffic in mobile networks could be established without any need for extra processing tool.

In [29] C. Huang et al. analyzed internet traffic data gathered from CDRs in Italy. They performed comparison of deep networks and non-deep networks over spatial and temporal features of the data. The models they utilized were RNN, 3d CNN, combination of RNN and CNN, ARIMA, non-deep NN. Multitask learning (MTL) approach was a combination of feature extraction and multi-task regression stages. The purpose of the work was to find minimum, average and maximum traffic value of network in next hour with changing intervals. As the result, CNN-RNN(MTL) model was chosen as all around model for forecasting traffic and the task of finding average usage performed better on all approaches.

2.3 Cluster Based LSTM Networks

In their work [30], Shao et al. proposed a multi branch LSTM model for stock price forecasting on the closing price data of Ping An Bank. Their model was first splitting time series data into equal length of sub-sequences. Then using K-Means clustering algorithm, they grouped the sub-sequences. For each group, they produced an output using LSTM network for forecasting. Their method had better accuracy over single LSTM model and BP Neural Network.

S.Vadyala et al. developed K-Means-LSTM model for the predictions of confirmed COVID-19 cases in Louisana state USA [31]. The model used clustering algorithms

for the day selection phase and applied different LSTM models for forecasting on each cluster. The proposed model had higher accuracy in terms of RMSE compared to base statistical method,SEIR.

In [32], J.Zhang et al. developed a cluster-based LSTM (CB-LSTM) network for short-term passenger flow forecasting in rail transit in China. The model first clustered the data according to ridership volume and passenger flow. Then, the CB-LSTM model was utilized in predicting short-term passenger flow. The first clustering method was combined with a predictability assessment model before using the novel LSTM model. Results showed that proposed model was superior to different statistical and deep learning methods under different measures.

2.4 Summary of Related Work

Time series analysis forms an important field in machine learning with the developments of processing power and increment of data in almost every area. As it can be seen in literature overview, deep learning models begin to substitute traditional statistical models where forecasting plays an important role for decisions in areas like financial and economic data, computer network traffic data. Together with these data, our research shows that CDRs are among one of the most crucial data for city planners or service providers to predict especially next location of the users. As we gather from our research, deep learning models like ANNs and RNNs are effectively being used for the analysis of CDR data.

Cluster based models in LSTM networks are relatively new in the literature. These works mentioned in Section 2.3 form an inspiration for this thesis. With the combination of these ideas, we expand to different aspects for improving LSTM network results.

CHAPTER 3

BACKGROUND & DATASET REPRESENTATION

As examined in Chapter 2, various methodologies are used for studying time series and many different techniques that are utilized for exploring CDR data bring favorable outcomes to the literature. We handle the problems defined in Chapter 4 as deep learning problems. Therefore, we need to elaborate the basics of models that have been constructed for analyzing the problems. After making the reader familiar with the basic concepts of deep learning methodologies that we have resorted in this work, we describe the time series clustering methods that make the models perform better in two of the problem sets. At the end of the chapter, we explore the dataset with describing columns and giving example rows.

3.1 Artificial Neural Networks

The perceptron, which was first proposed by F.Rosenblatt [33], is the atomic unit of artificial neural networks. In a basic form of nerve cell (neuron), as dendrites collect signals and axon spreads the integrated signals to other neurons [34], the perceptron is an inspiration of this structure. The output of a perceptron is the summation of weighted inputs [35], which can be formulated as:

$$o = \sum_{i}^{N} w_i x_i \tag{31}$$

A non-linear activation function can be applied for scaling output. Sample perceptron structure is shown in Figure 3.1.

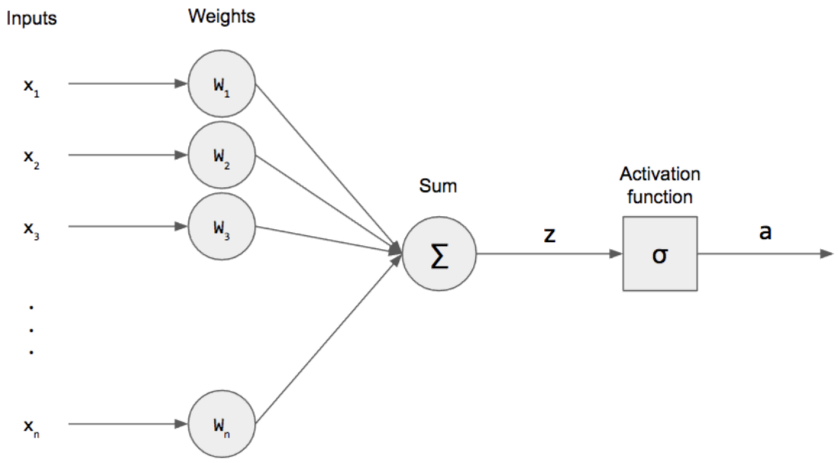


Figure 3.1: Structure of a Perceptron[1]

3.1.1 Multilayer Perceptrons

MLP is a type and more complex form of perceptron model. It is composed of different number of hidden layers stacked on top of each other with an input layer and also an output layer. Layers are comprised of different number of perceptrons and activation functions. If the number of hidden layers is more than one, than this structure is called a deep network [35]. A demonstration of fully connected feed-forward neural network is shown in Figure 3.2.

MLPs are used for many tasks of classification [36], [37], [38] and regression[39], [40], [41]. In general, if the output layer of MLP is composed of one perceptron, then the model is used for regression analysis. Classifiers like k-nearest neighbour, SVM or softmax evaluate the performance of neural network in the output layer for the classification tasks. The loss from output layer is back propagated through the hidden layers and the weights are updated with gradient descent algorithm [42]. This process enables the network learn the patterns according to given inputs and their corresponding output. The purpose of the gradient descent algorithm is thus to reach the global minimum of weight matrices that perform best results according to losses in the output layer.

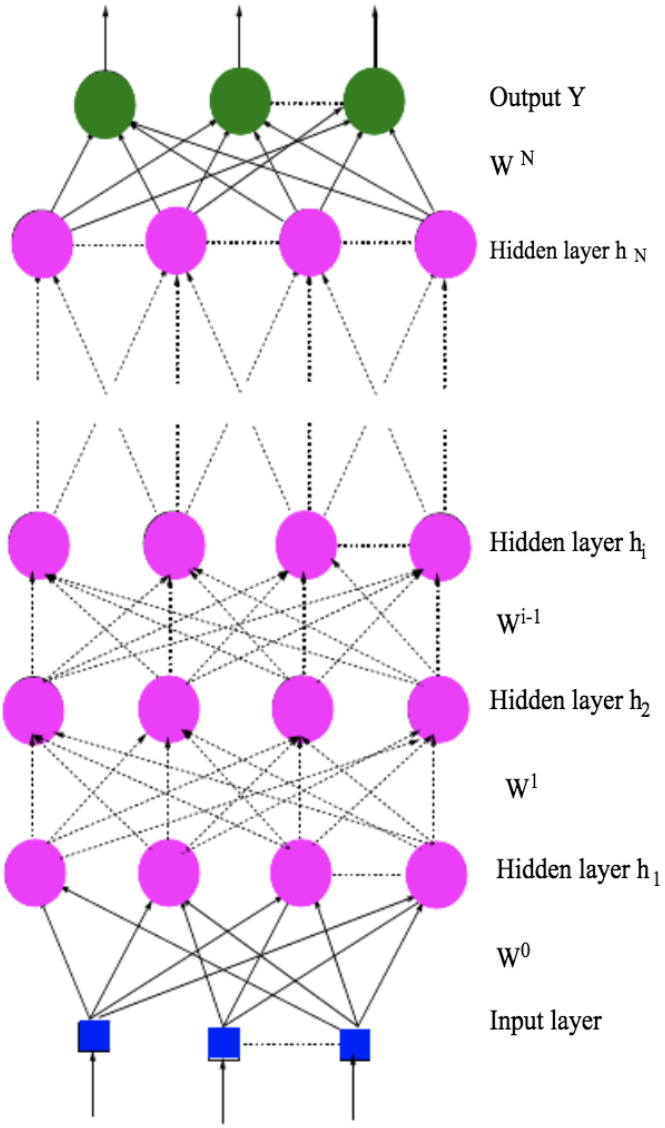


Figure 3.2: Structure of a Multilayer Perceptron[2]

3.1.2 Recurrent Neural Networks

Unlike feed-forward neural networks, Recurrent Neural Networks are designed as a solution for handling sequential input variable in length or producing output in variable size. According to their input and output size, RNNs can be categorized as many to one (sentiment analysis) [43], one to many (image captioning) [44], many to many (machine translation or video classification) [45] RNNs.

RNNs are good at holding and capturing the information of previous time steps thanks to its hidden states which learn from the inputs that rely on previous context of the data. A folded and unfolded depiction of a basic RNN is shown in Figure 3.3.

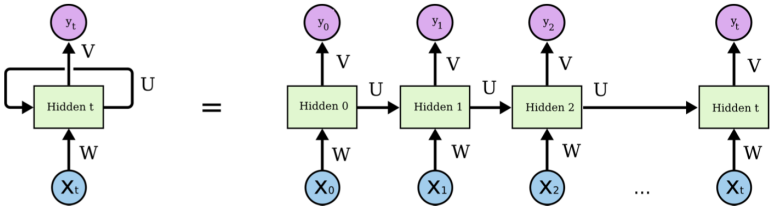


Figure 3.3: Structure of RNN[3]

In a simple form of RNN, there is an input layer $\mathbf{X_t}$, an output layer $\mathbf{y_t}$ and a recurrent layer $\mathbf{h_t}$. $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{V}$ are the weight matrices that map the dependencies between input layer, time steps and output layer. The mapping functions and forward propagation of RNN can be formulated as follows:

$$h_1 = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}_h) \tag{32}$$

$$h_i = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{U}\boldsymbol{h}_{i-1} + \boldsymbol{b}_h) \tag{33}$$

$$\hat{\mathbf{y}}_i = softmax(\boldsymbol{V}\boldsymbol{h}_i) \tag{34}$$

where $\boldsymbol{b}_h$ denotes the bias vector, σ is a type of an activation function. The purpose of training in the network is to find the optimal weight matrices that minimizes the loss of the network.

3.1.3 Back Propagation Through Time and Vanishing Gradient Problem

Back Propagation Through Time (BPTT) [46] is generalized form of back propagation, which is used in RNNs for sequential data. The loss is calculated in every time step for output. The weight matrices, which connect inputs to hidden layers and hidden layers to output sequences, should be updated according to accumulated loss of overall network. The simplest representation of BPTT is shown in Figure 3.4.

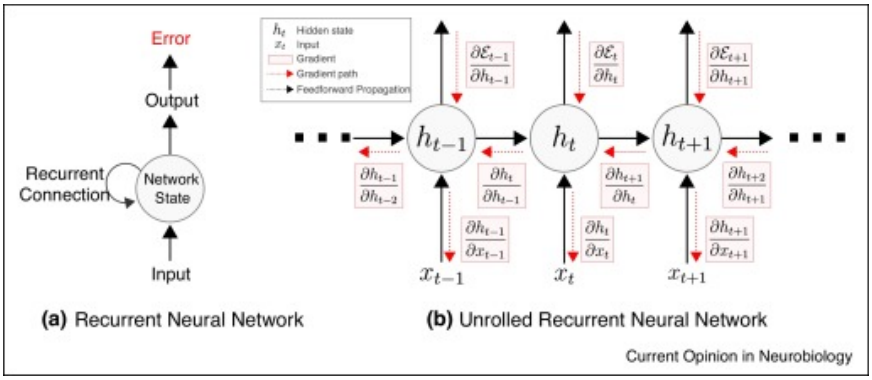


Figure 3.4: Representation of BPTT[4]

In vanilla RNN architecture, at every time step during back propagation, multiplication of some part of weight matrices cause gradient flow to be many factors of scalers. If the weights are between 0 and 1 this multiplication causes the weight update to be vanished through sequence of updates. If the weights are greater than 1, on the other hand, this will cause the weights to blow up after many time steps. These problems are called Vanishing or Exploding Gradient Problems, respectively. This unstable error updating structure of RNN makes the learning process of the network for the long term dependencies inefficient [47].

3.1.4 Long Short-Term Memory

Long Short-Term Memory (LSTM), which was first proposed by S. Hochreiter and J. Schmidhuber, is a type of recurrent unit that overcomes gradient updating problem [48]. Its solution to gradient problem makes the network learn long term dependencies.

Like perceptrons in ANN, LSTMs have cell mechanisms which are called as gates. These gates provide the flow of incoming and outgoing information. Although, there are many different representations of LSTM cell, it is mainly comprised of **input**, **forget** and **output** gates. A sample representation of LSTM cell is shown in Figure 3.5.

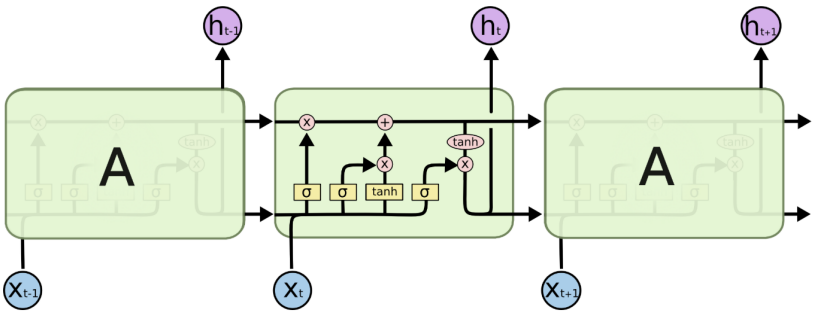


Figure 3.5: LSTM Unit[5]

As seen in Figure 3.5, unlike vanilla RNN structure, LSTM has an extra state, called Cell State (C_t), which makes the hidden state (h_t) of the cell gain or lose information as an output. C_t is summed function of input and forget gates, whereas h_t is a multiplied function of output gate and C_t.

Input and forget gates can be formulated as:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{35}$$

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{36}$$

Candidate values that will be added to cell state can be given as:

$$\tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{37}$$

Then, cell state is formulated as follows:

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t \tag{38}$$

The output gate that decides which part to output is given by:

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{39}$$

Hidden state finally can be calculated as:

$$h_t = o_t \otimes tanh(C_t) \tag{310}$$

The update process of C_t is composed of some element-wise multiplication and addition of gate operations. This provides clean (not oscillating) update, which is different at every iteration, of gradient flow in back propagation process. The gradient flow thus makes the network learn long term dependencies.

3.2 Clustering Algorithms

Unsupervised learning is one type of machine learning algorithms in which data points are unlabeled or unclassified. Thus, it is a technique for finding patterns in data sets according to similarity of the data points to each other [49]. It also includes dimension reduction of the features in the data set. Clustering, as a part of unsupervised learning, is the process of grouping similar data points according to some unlabeled features or patterns [50].

As an example of high dimensional data, time series need more detailed analysis for clustering. The purpose of clustering in time series is to find similarity according to a distance function. Two time series Q and C, where $Q = q_1, q_2, ...q_n$ and $C = c_1, c_2, ...c_n$ are similar if $dist(Q, C) \leq \epsilon$, where ϵ is defined threshold to measure similarity [51]. For clustering time series, one approach can be modifying the similarity measures and finding the best one according to the data set. Another option is to treat N data points in time series as N different features and then apply classical clustering methods.

Clustering methods are traditionally divided into two groups as **hierarchical** and **partitioning**, although there are many overlapping subgroups and different approaches for classifying the clustering algorithms [52]. However, in this work, we use **partition relocation** and **density-based** methods in the data-preprocessing part of the experiments. As partition relocation method, we utilize K-Means with Euclidean Distance and DTW. For density-based method, we cluster the data using DBSCAN algorithm. In the following subsections, we describe and analyze these methods.

3.2.1 K-Means Clustering Method

As one of the most popular clustering methods, K-means algorithm tries to reach to minimum error between each instance and randomly defined centers called **centroids** [53]. The algorithm starts with user-defined k numbers of centroids. After initialization of centroids, the algorithm calculates the distance of each instance to the centroids. Then, every instance is reassigned into a cluster according to the distance to the nearest centroid. This two-step iteration repeats until convergence is found.

Euclidean Distance, as one of the measures to find the distance between two data points p and q, is given in the Equation 311.

$$dist\left(p,q\right) = \sqrt{\sum_{i=1}^{n}\left(q_i - p_i\right)^2} \tag{311}$$

With the usage of Euclidean Distance, Algorithm 1 finds the suitable k clusters, where k is randomly selected [54]. Despite its popularity, K-Means Algorithm with Euclidean Distance for clustering time series have some disadvantages [7]. Firstly, Euclidean Distance measure needs time series of equal length. Secondly, it is not robust to outliers or noise. Thirdly, different transformations of time series make the measure very sensitive and thus present pessimistic similarity measure.

3.2.2 Dynamic Time Warping

For the drawbacks of Euclidean Distance (ED) measure, we use more matching and robust distance measure in K-Means algorithm for time series clustering part of the analyses. Dynamic Time Warping method [55] makes **many-to-one** or **one-to-many** comparison instead of point-to-point matching, therefore solves the requirement of 'equal length' in time series, and also detects the transformations like shifting or scaling by finding optimal path between time series [7].

Figure 3.6, as an example, shows the different clusters that come out of Euclidean and DTW distances. DTW distance manages to shorten the distance between Time Series-1 and Time Series-2. Besides, it places Time Series-4 and Time Series-5 into same cluster, finding the similarity between them.

Algorithm 1 K-Means Algorithm

Input
    data, k, centroids, clusters
Output
    clusters
repeat
    Define randomly selected $kcentroids$
    for $i\ in\ range(data)$ **do**
        $closestCentroid \leftarrow 0$
        $minDistance \leftarrow$ maxValue
        for each $centroid\ in\ centroids$ **do**
            $distance \leftarrow \|data[i] - centroid\|$
            if $distance < minDistance$ **then**
                $minDistance \leftarrow distance$
                $closestCentroid \leftarrow centroid$
            end if
        end for
        $data[i].cluster \leftarrow closestCentroid$
    end for
    Update $centroids$ with new $clusters$
until Convergence is reached

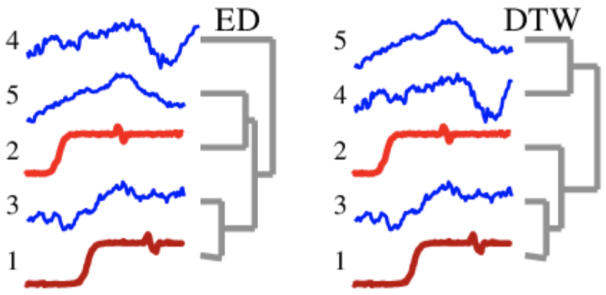


Figure 3.6: Clusters with ED vs. DTW [6]

In Figure 3.7, green lines represent the matching path between time series S and T. When two time series are inspected, we see that oscillations are matched in DTW distance, yielding shorter distance. The figure is also an example of how DTW compares time series with different lengths.

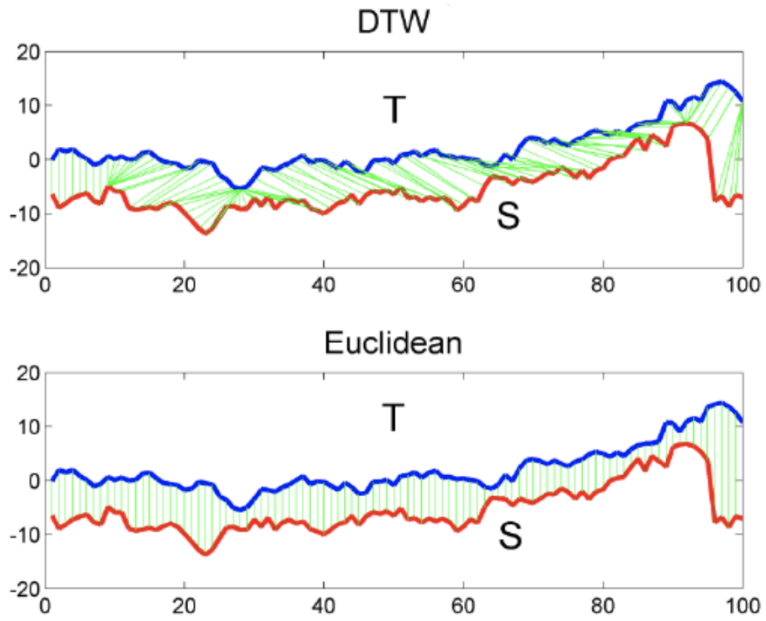


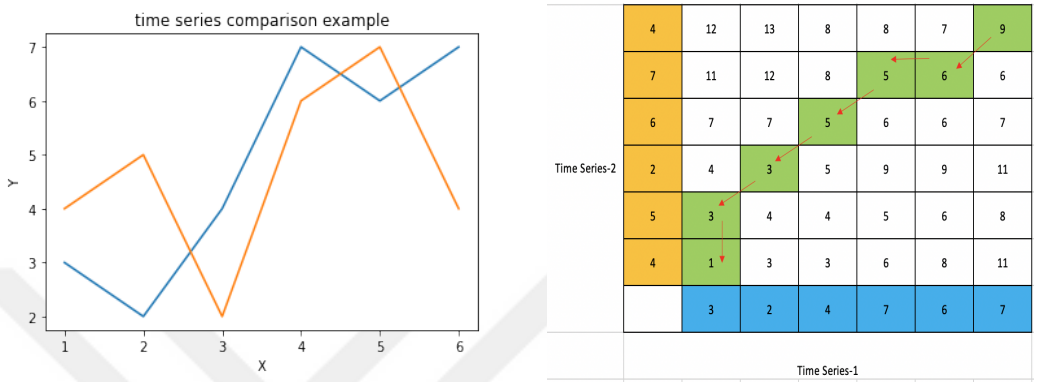
Figure 3.7: Many-to-One or One-to-Many Matching (DTW) vs. Point-To-Point Matching (ED) [7]

DTW distance between two time series, $P = p_1, p_2, ..., p_m$ and $Q = q_1, q_2, ..., q_n$, is calculated with the help of Cost Matrix. Cells in the matrix C are generated with the Equation 312 [56]. The cell in C(n,m) shows the cumulative distance between the series. The value of warping path, W, which shows the mapping between P and Q is the result of minimization function given in Equation 313. In here, K shows the length of warping path up to point(i,j) and is calculated using dynamic programming.

$$C(i,j) = dist(P_i, Q_i) + min[C(i-1, j-1), C(i-1, j), C(i, j-1)] \quad (312)$$

$$DTW(P,Q) = min \sum_{k=1}^{K} W_k \quad (313)$$

Figure 3.8a shows two example time series, Time Series-1 and Time Series-2, and DTW cost matrix with the warping path between these two time series is presented in Figure 3.8b. The linearity in the warping path is the signal of similarity between the series. Diagonal movement in the path shows that series are equal during that interval. Horizontal movement in the path means acceleration in the compared sequence, Time Series-2, while vertical movement shows deceleration in the particular interval.



(a) Example of Two Time Series

(b) DTW of the Time Series with Warping Path

Figure 3.8: Example of DTW Method

3.2.3 DBSCAN Algorithm

DBSCAN, which stands for Density-based Spatial Clustering of Applications with Noise, is one type of **density-based** clustering algorithms, which is proposed by Ester et al. [57]. As the name suggests, it is an algorithm which detects noise or outliers in large data sets.

The algorithm needs two defined parameters, which are ϵ and $MinPts$. ϵ is the distance used for limiting the distance within points in the same cluster. $MinPts$ is the minimum number of neighbor points to integrate the point into cluster.

The algorithm works in the following way. After defining ϵ and $MinPts$ parameters, a random starting point is firstly selected . If there are $MinPts$ within ϵ distance, then this point is defined as **core point** and the cluster is constructed. If points in **neighborhood** are also **core points**, then these neighbor points and holding **border**

points expand the cluster. After this iteration, another point that is not visited before is selected and the same procedure is applied. This procedure goes on until there are no remaining points.

An example run of DBSCAN algortihm on a sample data set is shown in Figure 3.9. Main advantages of the algorithm are as follows:

- The number of clusters is not defined beforehand as opposed to K-Means algorithm,
- The notion of outliers are well defined and the algorithm finds outliers easily,
- The algorithm also detects arbitrary shapes of clusters.

One disadvantage of the algorithm is that choosing input parameter combination can be difficult. In addition to this, the order of selected points is critical for defining the cluster shapes.

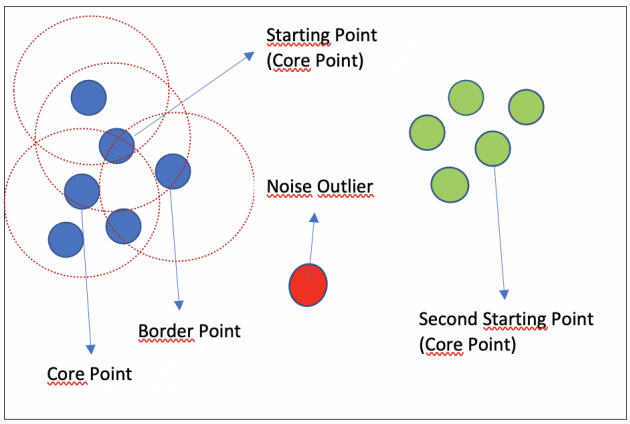


Figure 3.9: Result of DBSCAN on a sample data set

3.3 CDR Data and Representation

In the experiments part of this work, we utilize CDR of one of the well-known cell phone operators of Turkey. The CDR data includes almost 1 million users' log records that was produced for a period of one month. As it is concluded from region id section of data, the records are gathered from one of the most crowded cities of Turkey

(Ankara). For each user there are 30 records for each day on average and the region contains more than 13000 base stations.

Each row in data corresponds to a mobile phone transaction which can be either a phone call, an SMS (Short Message Service) or a GPRS connection. The columns of data correspond to eleven attributes listed below and the properties of them are as follows:

- **Cell Id-1:** Connected base station id by the caller or transactor.
- **Phone Number-1:** Phone number of caller or transactor. It is a hashed value because of security concerns.
- **City-1:** The city number of caller or transactor. Since one city is selected for producing the data, it is same for all users.
- **Cell Id-2, Phone Number-2, City-2:** Except that these three attributes are for callee, they have the same properties with the attributes corresponding to the first user.
- **Date of Action:** A string value that shows the action date.
- **Time of Action:** A string value that shows the action time.
- **CDR Type:** Attribute corresponding to category of action, which can be incoming or outcoming call(‘mmt ‘or ‘mmo’), sending or receiving SMS(‘msmo’, ‘msmt’) or GPRS connection(‘gprs’).
- **url:** It represents the gprs data’s url and it is available only for gprs actions.
- **Duration:** It represents the time length of action and not available for SMS actions.

An example row of CDR data is given in Table 3.1.

Table 3.1: One Example Row of CDR Data

10787	71eb716c160d8781ee54fb80537f6ffa	6	10787.0	6fdeaff8a79fb4e562908086bc39b1f0	6.0	20120912	164651	mmt	NaN	27

CHAPTER 4

PROBLEM DESCRIPTIONS & MODELS & ERROR METRICS

4.1 Problem Descriptions

After detailed analysis of CDR data, we propose three main problems for evaluating the models. They are categorized as Regression and Classification problems, which are namely:

- **Call Count Problem** (Regression Analysis)
- **Call Time-Sum Problem** (Regression Analysis)
- **Next Location Problem** (Classification Analysis)

In the following subsections, we define and exemplify these three problems.

4.1.1 Call Count Problem

Daily call count of a person and its prediction from past data logs can be beneficial for service providers to make the base stations denser and to increase quality of service in crowded places. It is also one way for providing personal advice, advertisement recommendation or billing policy for telecommunication companies.

The model for **Call Count Problem** can be formulated as follows:

Let the call count of the specific time interval, $\boldsymbol{t}$, for user $\boldsymbol{i}$ be $\boldsymbol{N}$. The purpose of the models is to predict the call count in $\boldsymbol{t}+1$. According to data set, $\boldsymbol{t}$ goes from 1 to 30, if the day is not discretized. If the day is discretized to 4 parts, namely morning,

midday, evening and night, then $\boldsymbol{t}$ is between 1 and 120. As an example for time series, $\boldsymbol{S}$, of 100 users with non-discretized time intervals is denoted as following:

$$S = s_1, s_2, ..s_{100}$$
$$and$$
$$s_i = N_1, N_2, ...N_{30}$$

We divide our experiments into three categories for this problem. First we select the best individual user based model among three different LSTM models. Then, we experiment on the best model using K-Means clustering, with the distance metrics both Euclidean and DTW, in data preprocessing part. As the last experiment, we apply DBSCAN algorithm to the best model. The benefit of DBSCAN is that we detect outliers in the data set and after separating outliers, we get the results for the users.

After constructing the data to be suitable for the model, we get the predictions from applied model. Then, the errors are calculated from the predictions and real values. The flow chart of this process is shown in Figure 4.1.

4.1.2 Call Time-Sum Problem

Total duration of calls for each person can be beneficial in identifying over-use or fraud in mobile phone usages. Thus, alerts can be sent to users. Also, it can be critical in building customer profiles for different patterns of usage. Again, as it is in call-count problem, operators can detect and maintain services to specific locations or different groups.

Similar to **Call Count Problem**, given the sequences ($S = s_{u^1}, s_{u^2}, ..., s_{u^t}$) where S is the time series of call duration sums, u is the notation for user id and t is again between 1 and 30 for non-discretized models or 1 and 120, otherwise. The purpose of in this problem is to predict $s_{u^{t+1}}$. The flow chart in Figure 4.1 is again applicable in this problem, except that in data preprocessing part, we sum the duration of calls for each user and we fill Nan values as the mean of user's call time-sum . An example

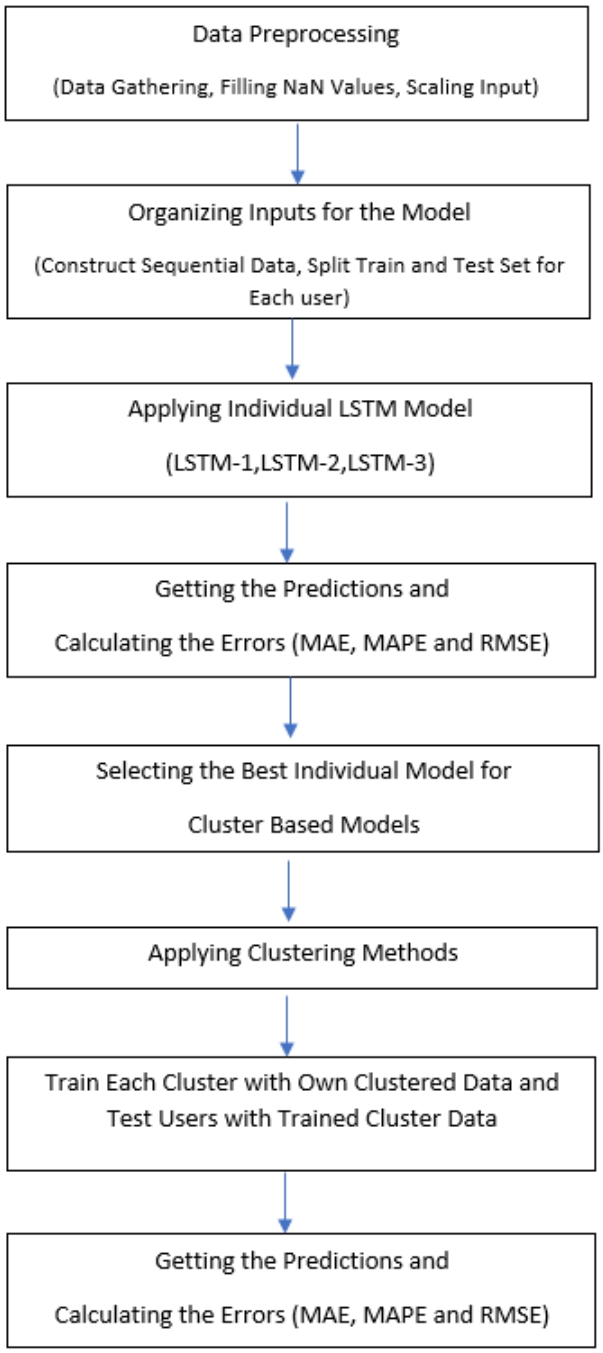
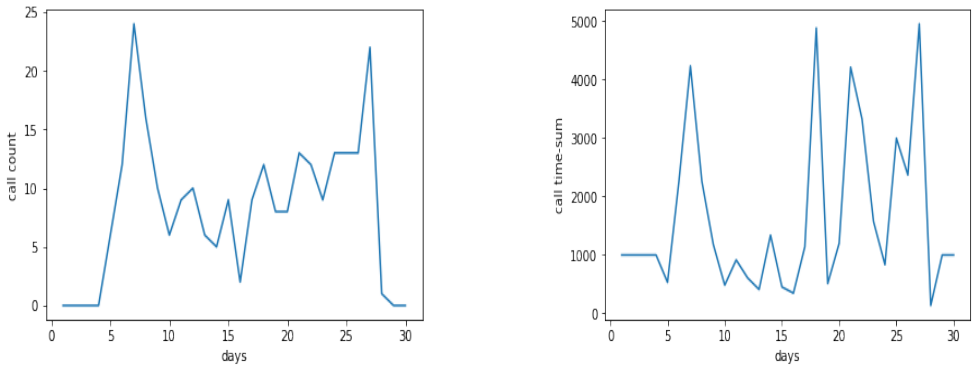


Figure 4.1: Flow Chart of Call Count Problem

time series for both call count and call time-sum is shown in Figure 4.2.



(a) Time Series of One Person(Call Count)

(b) Series of One Person (Call Time-Sum)

Figure 4.2: Time Series for One Person

4.1.3 Next Location Prediction Problem

Location prediction is an important data source for both governors and city-planners. The base station information of the users plays an important role for approximation of the callers' location. This information can be used in various areas like health, transportation, sales forecasting and recommendation etc. While city-planners can identify congested roads for reducing congestion in main roads, in case of a disease, governors can find out where the spread probability is higher in the city. Furthermore, with predicting the next location appropriately, operators can send catchy advertisements, which may be beneficial for both users and store owners.

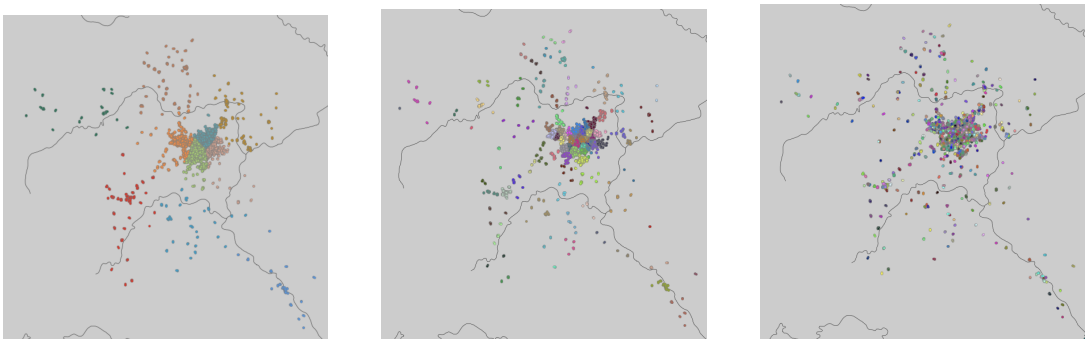
The purpose of the model is to find the next location with highest probability among all locations that each user visits. Given the location sequences of user u, ($l_{u^1}, l_{u^2}, ..., l_{u^t}$, we try to predict l_{u^t+1} and compare it with real location, thus yielding the accuracy of each user. In order to construct a location matrix, we define an amount of N locations (Average call count for each user group), trim the locations above margin and fill the empty values with last visited locations.

Figure 4.3 shows how we model the locations as inputs and targets. As it can be seen in the structure of the data, the sequential input is applicable for LSTM network that

we propose in following subsections.

Figure 4.3: Location Sequences and Prediction Strategy

Since the base stations are too close to each other in city center, we propose K-means clustering algorithm with Euclidean distance (Section 3.2.1), using latitude and longitude information of the base stations. For the applicability of the algorithm we choose 10, 100 and 1000 cluster numbers for approximately 13000 base stations. Figure 4.4, shows the groups of base stations that are clustered.



(a) 10 clusters (b) 100 clusters (c) 1000 clusters

Figure 4.4: Base Station Clusters

After clustering all of the base stations, the labels of them are fed into an embedding layer. The base stations can be regarded as words in language model. For each of the user, visited locations can be structured as sentences of locations. These sentences then can be used to predict the next word which is again a location of base station. We compare LSTM model with same structured MLP model and **the Most Frequent Place** model.

4.2 Models

In this section, we define the models that we use for the analysis of CDR data in different aspects. For the analysis of first two regression problems defined in Section 4.1.1 and 4.1.2, we use individual user-based models, cluster based models and outlier separation model with DBSCAN. For the classification task, namely **Next Location Problem**, we compare base LSTM model with MLP and Most Frequent Place model.

4.2.1 Individual User-Based Models

4.2.1.1 Base LSTM Model

In the base form of LSTM model, we divide time series of users to sequences. These sequential data are shaped as **samples, features and time-steps** for the input to LSTM blocks. Every LSTM block produces an output which is an ahead of time series sequence. Figure 4.5 shows this demonstration.

Between blocks of LSTM, we use **relu** activation function [58]. After preliminary trials for 100 users, we decide to use **adam** optimizer [59] for the updating policy of weights. In the classification task, we use the loss function of **categorical_crossentropy** while in the regression tasks, we use **mean squared error**.

Embedding layer is beneficial to represent the locations as dense vectors. With the power of this layer, sparse one-hot encoded vectors can be turned into dense vectors. The classification task, thus, takes the advantage of this layer for embedding long vectors. The embedding layer is combined with input layer in the next location

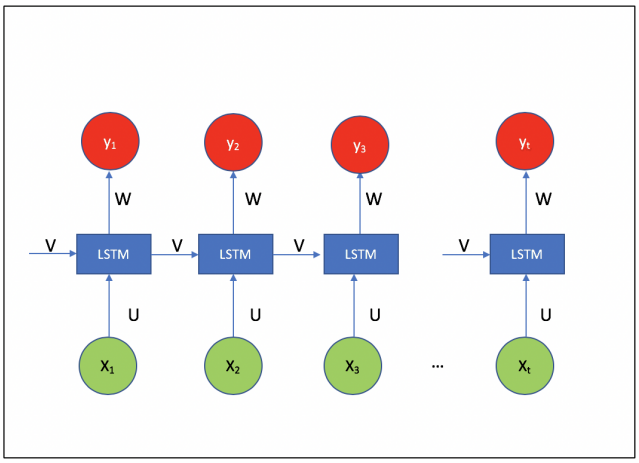


Figure 4.5: Base LSTM Model

problem.

In the model, 80% of the sequences of every user is selected as training and the remaining 20% is used for testing purpose. For the summary of prediction, we determine a model and apply it to every user. The errors of the users are gathered into arrays and mean of them is calculated. In this regard, this model is compared with other types of individual user-based LSTM models that we construct. Deep learning approach can be beneficial for finding non-linear structure of the data and this model comprises base for this purpose.

4.2.1.2 Batch Normalized and Dropout LSTM Model

Batch Normalization, which is first proposed in [60], is a technique used for both speeding the learning of neural network and making the network more stable regularizing the weight parameters. Although, the reasons behind how this concept is affecting the performance of neural network are under discussion, weight standardization in deep networks clearly reduces generalization error and speeds up the learning with smaller parameters, thus yielding lower epoch counts for reaching the global minimum in gradient descent algorithm.

Batch Normalization proposes a method for fixing every neuron to normalized zero mean and unit variance. Every input dimension is transformed to new scaled ver-

sion, with subtracting mean of mini-batch, $\mathbf{B}$, and divided to variance of $\mathbf{B}$. The transformation step then follows as:

$$y_i^{(k)} = \gamma^{(k)} \hat{x}_i^{(k)} + \beta_i^{(k)} \tag{41}$$

where $\hat{x}$ is scaled input, k is input dimension, i is mini-batch's instance. γ and β are the parameters that are learnt through optimization.

Dropout[61], as the name implies, is the technique for ignoring some of the neurons during training phase in both forward and backward pass. The reduced network, thus learns more generalized form of network which keeps away from over-fitting. The neurons are ignored with the probability of $1 - p$ with p comes from *Bernoulli* distribution.

In our empirical studies, we benefit from both of these techniques. The recent study in [62], shows that there is a disharmony between these two concepts. Considering this study, in the model, we use batch normalization with little dropout rate so that effect of dropout is less in the batch normalization part. Figure 4.6 shows the layers constructed for utilizing both of these techniques.

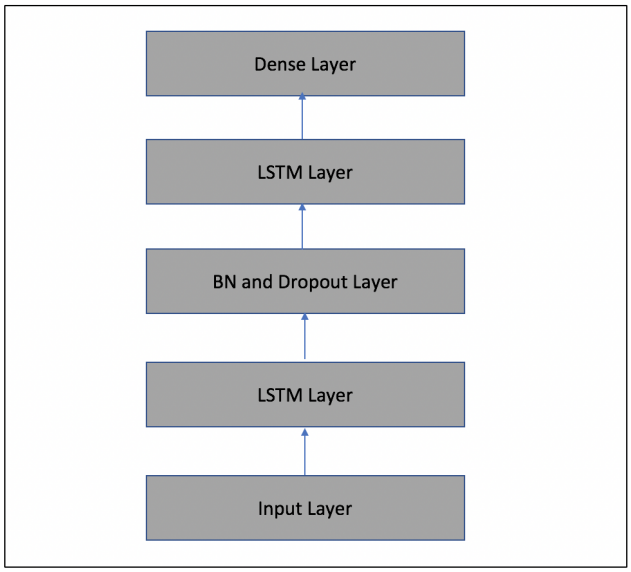


Figure 4.6: BN and Dropout Model

4.2.1.3 Multi Fully-Connected LSTM Model

In this model, we investigate the effect of stacking multiple fully connected layers on top of lstm layers. The idea behind this investigation is to see the impact of increasing the degree of polynomial that produces the output. Multiple dense layers are combined with activation functions in this type of model. Increasing the depth in the network clearly makes the model learn more complex features. However, every user has different patterns in terms of call count and call time-sum. The effect of this deeper model is investigated in the experiments part. Figure 4.7 shows the representation of this model.

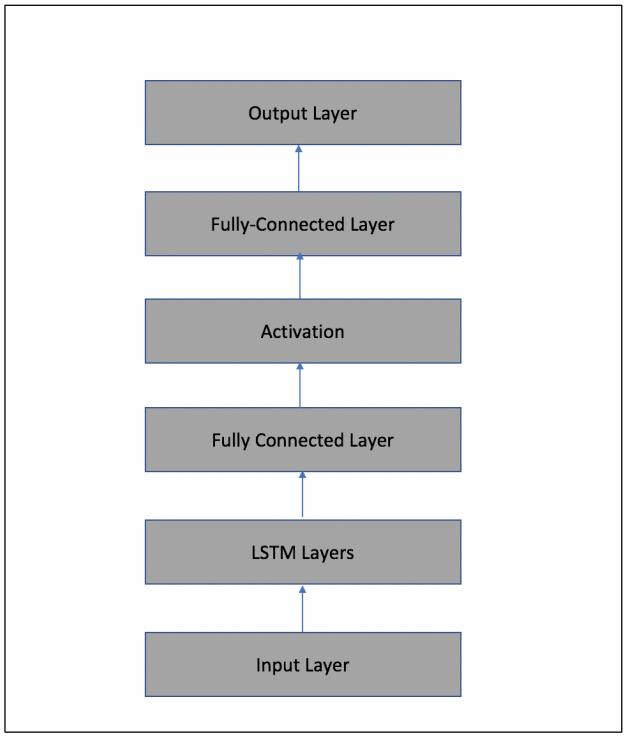


Figure 4.7: Multi Fully-Connected LSTM Model

4.2.2 Cluster Based Models

The intuition behind cluster based LSTM models is to see the effect of grouping similar behaviors of users. Since we have 30-day data for each user, we expect to see that increasing data amount for training with the help of clustered data provides

generalization to regression problems. Training each cluster with own data takes the advantages of handling similar behaviors of neighbor users in the cluster, so that test user predicts according to approximate average of train data. With the help of partitioned and increased data, we try to both handle unexpected call count or call time-sum actions of users and smooth the instant increases or decreases in learning process of clusters. Main disadvantage of clustering in data preprocessing part is that it produces overhead in terms of time.

In this model, we first cluster data according to K-Means algorithm with two distance metrics in time series of 100, 1000 and 10000 users, respectively . Then, for each test group, we select best individual model. We train **k** (cluster number) numbers of same LSTM model with each cluster data. After training process is completed, we test each user with trained cluster model. Figure 4.8 shows the general steps of these models.

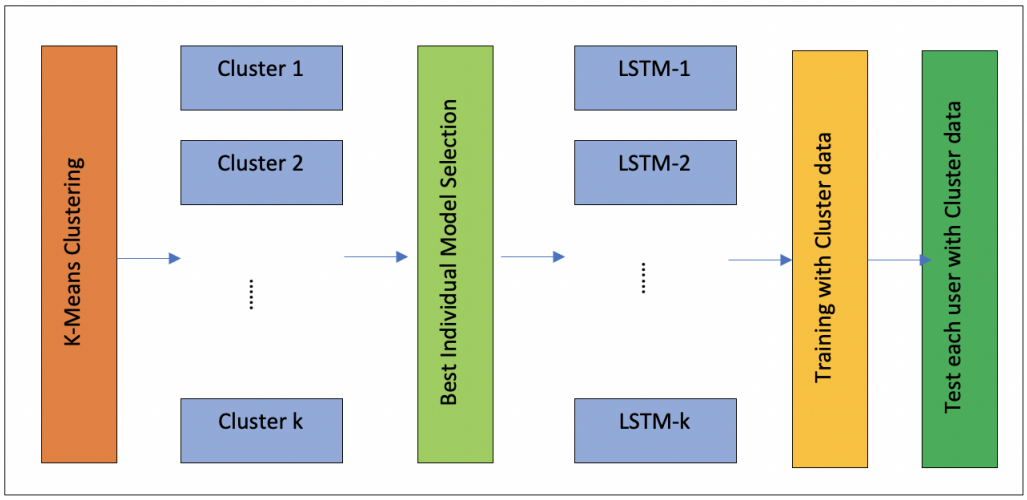


Figure 4.8: Layers of Cluster Based Model

For the purpose of increasing data amount, we train with all of user group data, for producing one main cluster. Thus, we generalize all of users into one group. In the experiments part, we see the effect of this approach.

4.2.3 Outlier Separation Model with DBSCAN

Main purpose behind selecting DBSCAN algorithm for clustering is to recognize noise in user groups and separate them from clusters. Thus, we make the clustered users closer in terms of behaviours. We try to reach to an optimal point for data agglomeration and decrease in unexpected clusters. Getting rid of outliers also accelerate learning process because of the decrease in data amount.

The steps for this model are as following:

- For finding best ϵ and $MinPts$ parameters, we experiment on different sets of the parameters
- After optimizing ϵ and $MinPts$ parameters, we cluster data and separate outliers
- We select best individual LSTM model
- We train on **k** numbers of same LSTM model according to data of each cluster
- We test on every user except for outliers.

Although we decrease the number of users and data for training, we expect to see more accurate predictions. The reason for this expectation is creating more meaningful clusters than K-Means clustering. As more advanced level of data decreasing, we see the effect of choosing only the most crowded cluster, where **k** equals one in steps of the model and the first five of most dense clusters, where **k** equals five. Thus, we only predict for most similar and lowest error rated users.

4.2.4 Base MLP Model

For classification task, namely next location problem, we construct the MLP model for the comparison to LSTM models. The inputs are fed into MLP model as location sequences. The benefit of this construction is that the same patterns of the users can be recognized by the model. We expect less success rate when compared to

LSTM models, since dependencies of the past values are not persistent for the next predictions.

Given the input sequence $l_1, l_2, ..., l_t$, this model predicts the value of l_{t+1}, where l denotes location. In the second iteration of prediction , given the locations $l_2, l_3, ..., l_{t+1}$, the model forecasts l_{t+2}. The locations are compared to ground truth locations and the accuracy of the model is calculated.

4.2.5 The Most Frequent Place Model

As a baseline model for classification task, next location prediction, we use a naive method for comparison to deep learning approaches.

For every user, we divide 30-day data as train and test split. We first select the most visited place, which is found from data records and connected base station of the record, in train set for each user. Then, we evaluate and compare the base stations of outgoing call (mms) to the selected and most frequent base stations. We expect to see good success rates for the users who do not change places very frequently. Thus, if the user group is mostly comprised of users with this kind of behavior, then the overall error rate becomes at a desired level.

4.3 Error Metrics

In this section, we explain the error metrics used for evaluating the models. We divide this section according to problem classes, namely regression and classification.

4.3.1 Regression Tasks

In order to evaluate the models used in the analyses of call count and call time-sum problems, we utilize RMSE, MAE and MAPE, respectively. RMSE is a metric which gives more penalty to outliers in prediction, when compared to MAE. MAPE, on the other hand, is the percentage equivalent of MAE. Table 4.1 gives the error metrics, where e denotes the error between predicted and real data points. For each user, we

find the values of aforementioned error metrics. After test process finishes and we calculate the errors, we take the average of all users yielding overall performance of the model.

Table 4.1: Error Metrics for Regression Tasks

Root Mean Squared Error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean Absolute Error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean Absolute Percentage Error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

4.3.2 Classification Task

In classification task, the next location problem, we benefit from accuracy metric, which is used generally in evaluating neural network performance. We evaluate the models of "The Most-Frequent Place", "Base Feed Forward Neural Network" and "LSTM Networks".

Accuracy can be explained in confusion matrix clearly. In confusion matrix, if the values predicted are True and they are actually True, they are called **True Positives(TP)**. If the values are again True, however, they are predicted as False, then they are called **False Negatives(FN)**. If the values are False and also predicted as False, they are **True Negatives(TN)**. The remaining part is called as **False Positives(FP)**. Table 4.2 shows this representation.

Table 4.2: Confusion Matrix

		Actual Class	
	Total	Yes	No
Predicted Class	Yes	TP	FP
	No	FN	TN

Accuracy, then can be formulated as following:

$$Accuracy = \frac{TP + TN}{Total} \tag{42}$$

In the **Next Location Problem**, we use the ground truth (real) places as True places and compare them with the predictions of the models. Therefore, for each user, accuracy of the model is calculated by:

$$Accuracy = \frac{Matching\ Number\ of\ Real\ Places\ and\ Predictions}{Total\ Number\ of\ Predictions} \tag{43}$$

After we evaluate every user's predictions, we measure the average of all of the users as model's performance.

CHAPTER 5

EXPERIMENTS

In this chapter, we respectively analyze the problems introduced in Sections 4.1.1, 4.1.2 and 4.1.3. For comparison of regression tasks, we first select the best individual user-based model for randomly selected user sets of 100, 1000 and 10000. We evaluate the performance of cluster based and outlier separation LSTM models according to selected base model. We make the model selection according to lowest score of RMSE error metric. In the classification task, we analyze the results according to cluster labels. Results are organized according to 10, 100 and 1000 clusters of base stations. We show the performances of 100, 1000 and 10000 users with respect to clusters. The accuracy results show the success of each model. We compare **The Most Frequent Place** model with **MLP** and LSTM models. We present detailed analysis at the end of each section.

The experiments are conducted on a MacBook Pro with a 2.7GHz dual-core Intel Core i7 processor, 16GB memory and 512GB disk space. We implement each model and analyze the results using Numpy, Pandas, Sklearn and Keras libraries of Python.

5.1 Preliminary Experiments

Stationarity of time series forms basis for the models to be more predictive in forecasting. If a time series is stationary, it is more consistent through time and thus can be more suitable for conventional statistical models. If it is non-stationary, on the other hand, it needs to be done some transformations to make it stationary. Since stationary time series have constant mean and standard deviation and also does not have any seasonal trend[63], fitting a statistical model will produce more robust predictions

according to compatible past values.

One way to make assumptions for the stationarity of time series is to apply statistical tests upon the data. Unit root tests can be used to determine whether time series are non-stationary and have unit roots with null hypothesis. As an example of unit root test, we utilized Augmented Dickey-Fuller Test [64] for detecting the stationarity of the time series in the two regression problems. We decided upon a threshold of 0.05 for p-value to reject the null hypothesis. p-value below 0.05 was enough to reject the null hypothesis and thus the time series variable was defined as 'stationary', otherwise 'non-stationary'. Table 5.1 shows the results of the test.

Table 5.1: ADF Test Results for Regression Problems

Call Count Problem /ADF Test Results		
Users	Stationary	Non-Stationary
100	73	27
1000	727	273
10000	7974	3026
Call Time-Sum Problem / ADF Test Results		
Users	Stationary	Non-Stationary
100	86	14
1000	758	242
10000	7717	2283

According to ADF Test results, we applied ARMA and ARIMA models to the series and tried to reach to an optimal point for both of the regression tasks via changing **p, d and q** parameters. Before cluster based LSTM analyses, we compared the results of ARMA and ARIMA models to individual user-based LSTM models. We analyzed 1-day, 2-day and 3-day ahead predictions on the analyses of the regression problems. We also made discretization dividing the day into 4, 6 and 12 parts. Our preliminary works presented messy and also unimproved results for both of the tasks. Since changing the parameters of the models did not work for any further improvement, we changed the structure of LSTM models and approached the regression problems as cluster analysis.

In the classification task, on the other hand, we experienced that deep learning approaches produced better results compared to naive model. For increasing the suc-

cess rate of the LSTM model, we also got results for **two prediction at once** LSTM models.

5.2 Analyses of Call Count Problem

In **Call Count Problem**, we first construct a generalized individual model for the analysis of each user's call count pattern. Then, we apply this model to user's data. %80 of each user's data is selected as training and the remaining part is used for test purpose. We place each user's results into specific error arrays and the mean of every error array shows the performance of the individual user-based model.

The best individual model forms basis for the cluster based models. We experiment on two different distance metrics for K-Means clustering and see the results on the numbers of 3, 5, 7 and 15 clusters. After cluster parameter selection phase, we experiment on cluster based data training and all of the data training. Our experiments for each subsection ends with DBSCAN clustering analysis. After finding the best parameter pair, ϵ and $MinPts$, for the user-group data, we experiment on outlier separating, most crowded and most five crowded cluster analysis. We divide overall analyses into two groups, Non-Discretized and Discretized time intervals.

5.2.1 Non-Discretized Time Interval Analysis

We start experiments with finding best parameter settings for base LSTM model, namely LSTM-1. We find the settings according to 100 users and then expand the same settings for 1000 and 10000 users. The parameters we change are listed below:

- **Sequence Length**: The length to use for input in time steps of specific user,
- **Epoch**: The number of times that LSTM model walks through the entire data set,
- **LSTM Cells**: The number of LSTM units in the specific layer,
- **Optimizer**: The method to use for changing attributes (weights, learning rate, etc.) of the network to reduce the loss in the model,

- **Loss Function**: The error function to use for estimating the loss of the model in one evaluation of the input and the output,
- **Learning Rate**:The hyperparameter to adapt the model to the errors and the measure for update policy of the weights of the network,
- **Batch Size**: The number of training examples for taking into account at one iteration for the evaluation of the model,
- **Activation Function**: The function to use for scaling summed inputs for a perceptron/node.

Table 5.2 shows the values and classes we change during the selection of the parameters. The best setting from the results of these parameters is as presented in Table 5.3.

Table 5.2: Parameter Values and Classes (Call Count/Non-Discretized)

Parameters	**Values / Classes**
Sequence Length	10, 15, 20
Epoch	10, 20, 100, 300
LSTM cells	4, 8, 16, 32, 64, 128, 256
Optimizer	SGD, RMSProp, Adam
Loss Function	MSE, MAE
Learning Rate	0.01, 0.001, 0.0001
Batch Size	8, 32, 128
Activation Function	Relu, Sigmoid, Tanh

Table 5.3: Parameter Settings for Individual User-Based LSTM Models (Call Count/Non-Discretized)

Sequence Length	Epoch	LSTM cells	Optimizer	Loss Function	Learning Rate	Batch Size	Activation Function
20	20	8	Adam	MSE	0.01	8	Relu

After choosing the best parameter settings, we apply them into 3 models, presented in Subsection 4.2.1. LSTM-1, LSTM-2 and LSTM-3 mean Base LSTM Model, Batch Normalized and Dropout LSTM Model and Multi Fully-Connected LSTM Model, respectively. Table 5.4 shows the results for Non-Discretized Call Count problem.

Analyses on RMSE and RMSE/Exp.Value show that LSTM-3 is the best model to apply into cluster based models.

Table 5.4: Individual User-Based Model Results (Call Count/Non-Discretized)

Users	100					1000					10000				
Error Metrics	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.
LSTM-1	7.483	4.396	5.076	153.017	0.67	7.457	4.020	4.501	128.455	0.60	7.458	3.992	4.453	130.360	0.60
LSTM-2	7.483	4.174	4.788	135.563	0.64	7.457	4.287	4.818	137.678	0.64	7.458	4.437	5.021	136.956	0.67
LSTM-3	7.483	4.130	4.737	137.258	0.63	7.457	4.010	4.496	126.779	0.60	7.458	3.976	4.438	127.776	0.60

We apply LSTM-3 model to different numbers of clusters and two types of distance measures for each user group. The best parameters are presented in Table 5.5. The table shows that the best parameters change with user group. We expand cluster analysis to have one cluster, which means we use for training all of the users' data in each user group. The results of clustered data and all data models, LSTM-C and LSTM-A respectively, are shown in Table 5.6.

Table 5.5: Results of K-Means Distance and Cluster Number Parameters (Call Count/Non-Discretized)

Users	100				1000				10000			
Cluster Numbers	3	5	7	15	3	5	7	15	3	5	7	15
K-Means (Euc.)	4.962	4.872	4.913	5.124	4.524	4.312	4.442	4.832	4.583	4.821	4.512	4.476
K-Means (DTW)	4.947	4.897	4.926	5.012	4.317	4.652	4.252	4.215	4.446	4.652	4.315	4.243

Table 5.6: Cluster-Based Model Results (Call Count/Non-Discretized)

Users	100					1000					10000				
Error Metrics	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.
LSTM-C	7.483	4.128	4.872	145.814	0.65	7.457	3.855	4.252	89.529	0.57	7.458	3.881	4.243	116.376	0.56
LSTM-A	7.483	3.865	4.606	131.523	0.61	7.457	3.939	4.338	103.468	0.58	7.458	3.456	3.857	109.449	0.52

The analyses continue with separating the outliers in each user group. The main purpose of the process is to make predictions for only reasonably foreseeable users. The parameters, number of outliers and number of clusters with respect to user groups are shown in Table 5.7. Figures 5.1, 5.2 and 5.3 show the distribution of the users and clusters. Red points indicate outliers in each group.

Table 5.7: Parameters, Outliers and Cluster Numbers for DBSCAN (Call Count / Non-Discretized)

Users	Eps. and MinPts Parameters	Number of Outliers	Number of Clusters
100	0.7 / 4	25	6
1000	1.2 / 4	243	107
10000	0.8 / 4	1778	1219

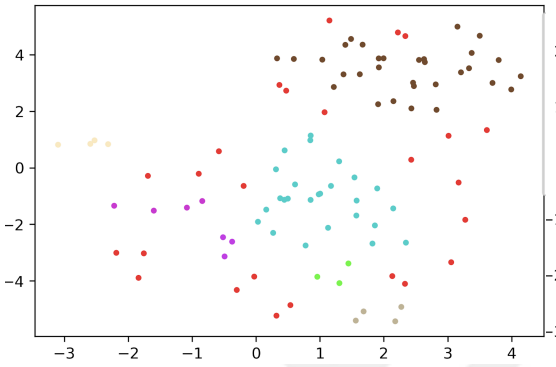


Figure 5.1: Clusters of 100 Users

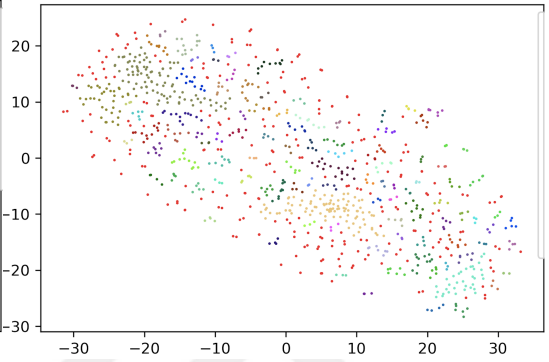


Figure 5.2: Clusters of 1000 Users

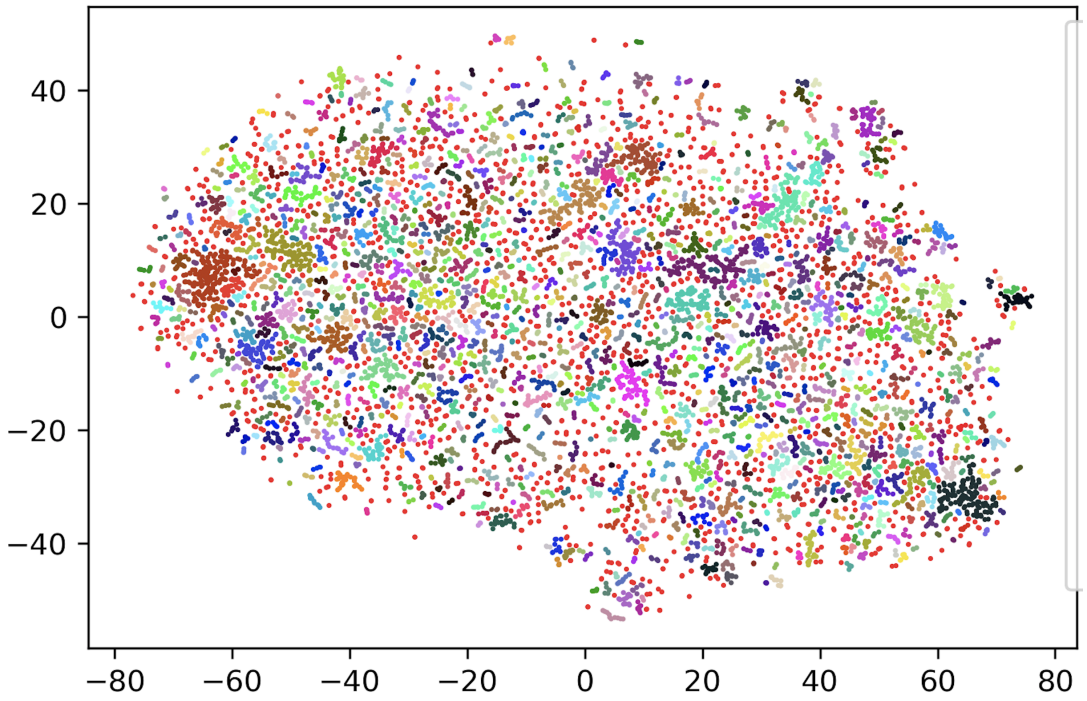


Figure 5.3: Clusters of 10000 Users with DBSCAN (Call Count / Non-Discretized)

LSTM-D-Gen. model means the clustered data only separating outliers. We make predictions only for the most crowded group and LSTM-D-Best model denotes that cluster. LSTM-D-Best-5 model, on the other hand, shows the predictions for the most five crowded groups. Table 5.8 shows the results of these models. We need to state the number of users we make prediction for each of DBSCAN model. Table 5.9 shows the amount of predictions for each user group.

Table 5.8: Outlier Separation Model Results with DBSCAN (Call Count/Non-Discretized)

Users	100					1000					10000				
Error Metrics	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.
LSTM-D-Gen.	10.256	4.128	4.478	74.710	0.43	8.722	2.019	2.182	54.369	0.25	8.222	2.432	2.596	72.573	0.31
LSTM-D-Best	8.758	3.549	3.883	79.158	0.44	11.370	3.307	3.318	62.754	0.29	9.761	2.272	2.540	68.034	0.26
LSTM-D-Best-5	9.832	4.039	4.419	86.488	0.45	6.852	1.683	1.807	86.533	0.26	9.639	4.048	4.446	100.329	0.46

Table 5.9: Number of Predicted Users (Call Count / Non-Discretized)

	Number of Predicted Users		
Models	100 users	1000 users	10000 users
LSTM-D-Gen.	75	757	8222
LSTM-D-Best	32	246	1738
LSTM-D-Best-5	68	538	4326

Figure 5.4 presents the results of all of the models for each user group. As a short summary, we clearly see the improvements in DBSCAN models in each user group.

Since we apply the same procedures on the following subsections of regression problems, we make the summary of the tables for being more integrated and present detailed analysis at the end of the subsections.

5.2.2 Discretized Time Interval Analysis

The main purpose of discretization of 30-day interval into 120 time intervals is to be able to catch specific behaviors of outgoing calls in morning, noon, evening and night times of users. Each day is separated to 4-parts, thus produces 120 time intervals for one user. We try to reach better error rates with discretization.

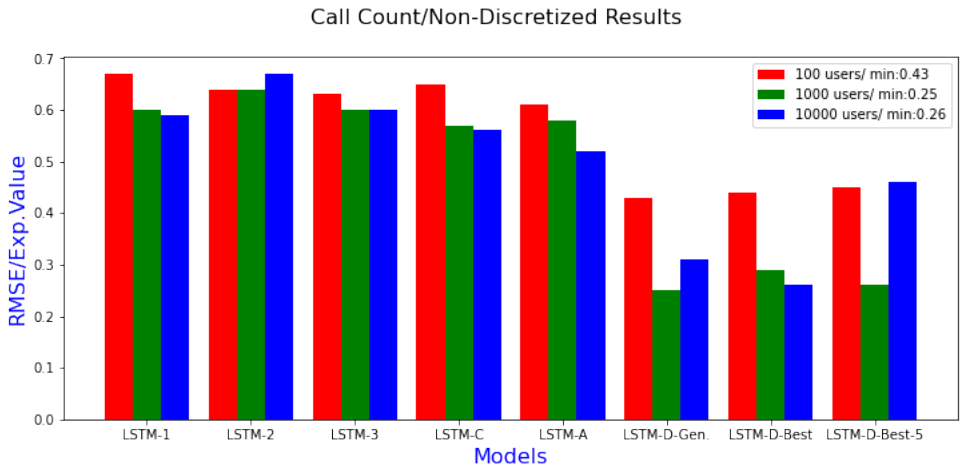


Figure 5.4: Overall Results of the Models (Call Count/Non-Discretized)

We start our experiments with selecting best parameter settings for LSTM-1. Table 5.10 shows the optimal values we reach after various experiments. With these settings, the results of individual user-based models are shown in Table 5.11.

Table 5.10: Parameter Settings for Individual User-Based LSTM Models (Call Count/Discretized)

Sequence Length	Epoch	LSTM cells	Optimizer	Loss Function	Learning Rate	Batch Size	Activation Function
100	100	32	Adam	MSE	0.001	32	Relu

We see that in majority of the groups, the best model is again LSTM-3. Therefore, we select it as a base model for cluster based analyses.

Table 5.11: Individual User-Based Model Results(Call Count/Discretized)

USERS	100					1000					10000				
Error Metrics	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.
LSTM-1	1.956	1.302	1.648	68.285	0.84	1.901	1.531	1.739	105.974	0.92	1.884	1.413	1.615	94.636	0.85
LSTM-2	1.956	2.014	2.180	157.176	1.12	1.901	1.322	1.644	73.68	0.86	1.884	1.331	1.616	83.872	0.86
LSTM-3	1.956	1.317	1.624	70.896	0.83	1.901	1.544	1.743	108.495	0.92	1.884	1.380	1.608	92.231	0.85

K-Means parameter selection is presented in Table 5.12. As the number of users increase, we observe that Euclidean Distance and higher cluster numbers produce better results.

Cluster numbers in Table 5.13 shows that the data depicts more compact behaviours

Table 5.12: Results of K-Means Distance and Cluster Number Parameters (Call Count/Discretized)

Users	100				1000				10000			
Cluster Numbers	3	5	7	15	3	5	7	15	3	5	7	15
K-Means (Euc.)	1.984	1.972	2.013	2.124	1.734	1.692	1.746	1.832	1.783	1.721	1.564	1.576
K-Means (DTW)	1.903	1.998	1.966	2.043	1.817	1.852	1.857	1.815	1.646	1.652	1.685	1.744

compared to non-discretized data. Besides, we see that outliers are less when data gets discretized.

Table 5.14 shows that we make predictions for high majority of user groups. We see the distribution of the users in Figures 5.5, 5.6 and 5.7.

Table 5.13: Parameters, Outliers and Cluster Numbers for DBSCAN (Call Count / Discretized)

Users	Eps. and MinPts Parameters	Number of Outliers	Number of Clusters
100	22.77 / 2	29	4
1000	1.19 / 4	139	42
10000	1.2 / 4	693	207

Table 5.14: Number of Predicted Users (Call Count / Discretized)

	Number of Predicted Users		
Models	100 users	1000 users	10000 users
LSTM-D-Gen.	71	861	9307
LSTM-D-Best	62	451	5472
LSTM-D-Best-5	71	583	6482

As the summary of results for cluster based models, we reach the same kind of improvement in DBSCAN models. RMSE / Exp.Value rates gets clearly lower in all of the outlier separating models. Table 5.15 gives the detailed results for each user group and cluster based models.

The summary of all models for discretized data in one picture is presented in Figure 5.8. We observe that main separation of outliers works for decreasing error rate.

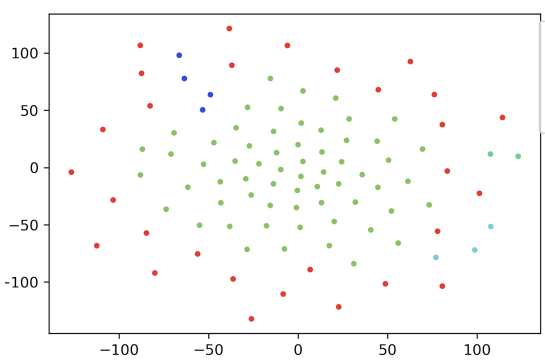


Figure 5.5: Clusters of 100 Users

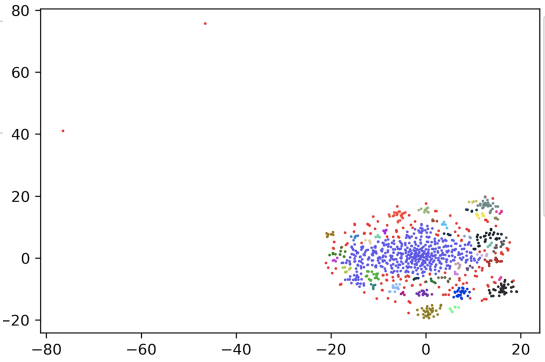


Figure 5.6: Clusters of 1000 Users

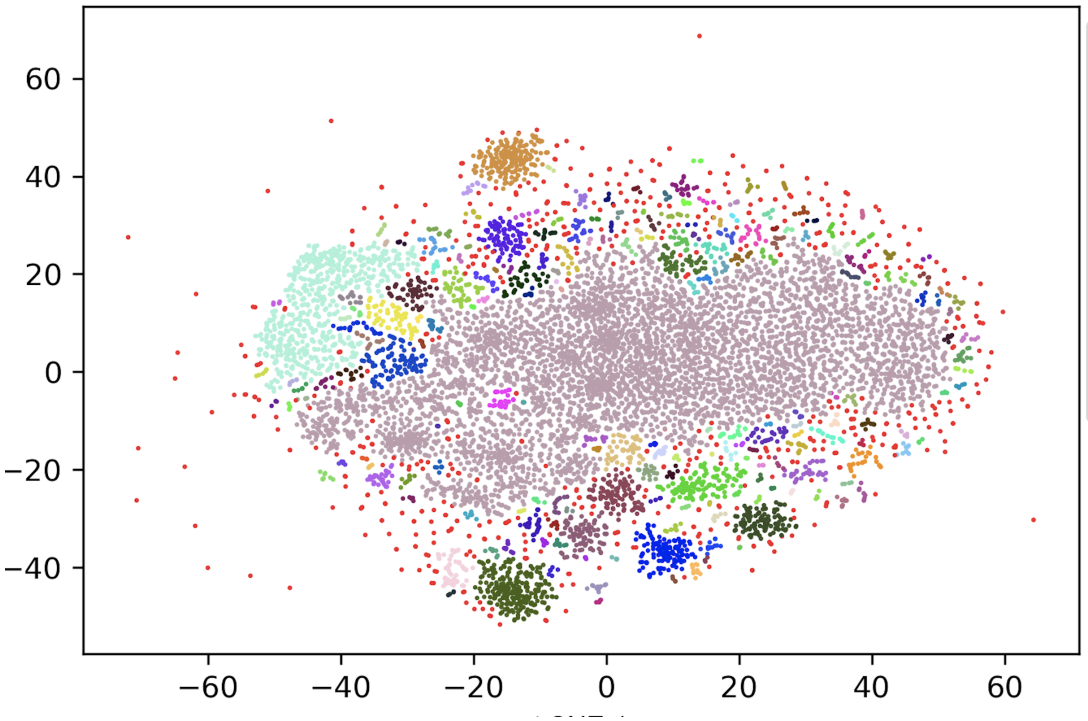


Figure 5.7: Clusters of 10000 Users with DBSCAN (Call Count/Discretized)

Table 5.15: Overall Results for Cluster Based Models(K-Means and DBSCAN) (Call Count / Discretized)

USERS	100					1000					10000				
Error Metrics	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.
LSTM-C	1.956	1.631	1.903	106.149	0.97	1.901	1.598	1.692	114.220	0.89	1.884	1.252	1.564	83.723	0.83
LSTM-A	1.956	1.844	2.084	133.577	1.06	1.901	1.306	1.602	68.527	0.84	1.884	1.042	1.363	63.168	0.72
LSTM-D-Gen.	4.357	1.357	1.454	64.980	0.33	4.299	1.297	1.587	58.641	0.36	4.297	1.230	1.568	55.960	0.36
LSTM-D-Best	4.247	1.360	1.748	63.238	0.41	4.243	0.921	1.237	48.247	0.29	4.147	0.982	1.217	46.481	0.29
LSTM-D-Best-5	4.177	1.358	1.763	63.791	0.42	4.326	0.925	1.278	45.058	0.30	4.321	1.223	1.521	53.250	0.35

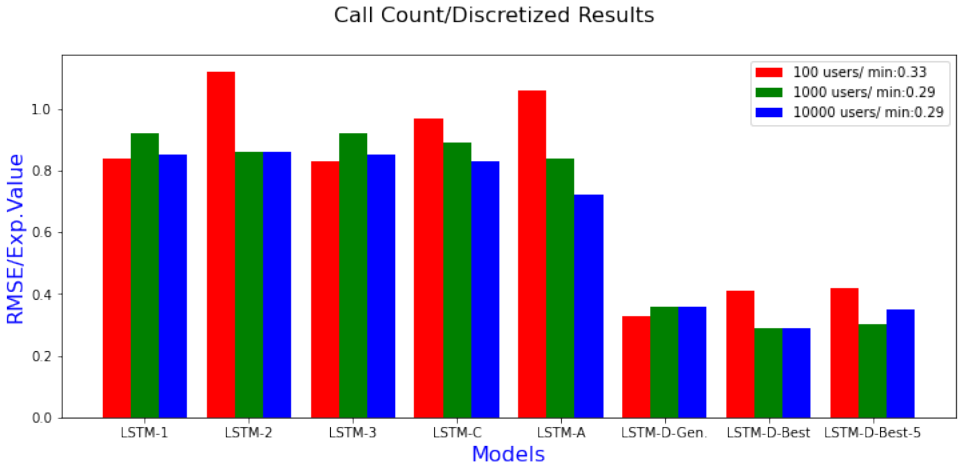


Figure 5.8: Overall Results of the Models (Call Count/Discretized)

5.2.3 Section Discussion

For both of the problem sections, we see that LSTM-3 model has better results in individual user-based models. Using Batch Normalization and Dropout has no significant effects for the improvement of the models. When we compare discretized data with non-discretized data for the call count problem, we find worse results in RMSE/Exp.Value rates for all of the user groups. At the same time, the error rates are not as expected for sequential data analysis using LSTM. For this reason, we change the approaches for the problems.

We start to see little improvements using cluster based LSTM methods in both of the problems. Especially, in Non-Discretized data, we clearly see the promising results with the increase in training data. LSTM-A model, in which we use one cluster, gives best results in cluster analysis as we increase the number of users and integrate them for one training data. This improvement gives rise to approach of separating undesired and unpredictable users and also cluster the remaining users. Meanwhile, since outgoing call behaviours of the users are distributed with heavy noise, the effect of selecting different distance measures in clustering is not clear.

For DBSCAN analyses, we see that separating outliers in discretized data causes more compact clusters when compared to non-discretized data. Outlier numbers are close to each other in both of the analyses. The main purpose for DBSCAN models is

to make predictions for the users having foreseeable behaviors as much as possible. Thus, as we increase the number of user groups, LSTM-D-Best model gets better error rates. The maximum improvement from the worst result is %61,20 at DBSCAN models in non-discretized data for 10000 users, while it is %66,28 for discretized data. We see the good scalability of DBSCAN methods as the amount of data increases and error rate decreases.

5.3 Analyses of Call Time-Sum Problem

In **Call Time-Sum Problem**, we sum outgoing call times of the users in time intervals. The data has measure in seconds at call times. %80 of each user's data is again selected as training in individual and cluster based models. The errors are averaged after finding each user's error.

We select the best individual user-based model for the model structures of clustered data. Selected K-Means cluster numbers are the same with **Call Count Problem**. Another cluster model is using all of data, LSTM-A. DBSCAN models finish each subsection analyses. We experiment for two subsets of data, Non-Discretized and Discretized time intervals.

5.3.1 Non-Discretized Time Interval Analysis

We start with the parameter settings for base LSTM model, LSTM-1. Table 5.16 shows the best results of various experiments on changing the parameters. We see the significant increase on epoch count and batch size. Besides, wider LSTMs of size 64 are showing better analysis results.

Table 5.16: Parameter Settings for Individual User-Based LSTM Models (Call Time-Sum/Non-Discretized)

Sequence Length	Epoch	LSTM cells	Optimizer	Loss Function	Learning Rate	Batch Size	Activation Function
20	100	64	Adam	MSE	0.01	128	Relu

Table 5.17 shows the results of individual experiments. We select LSTM-1 for the

base of cluster analysis and realize little differences as the amount of data increase with inspecting RMSE and RMSE/Exp.Value rates. Increasing fully connected layers at the end or adding Dropout and Batch Normalization layers have no significant effect on individual user-based models.

Table 5.17: Individual User-Based Model Results (Call Time-Sum / Non- Discretized)

USERS	100					1000					10000				
Error Metrics	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.
LSTM-1	1875.060	1152.335	1330.623	117.113	0,70	1891.104	1136.298	1324.493	130.017	0,70	1915.31	1201.936	1410.160	105.655	0,73
LSTM-2	1875.060	1204.435	1435.350	113.379	0,76	1891.104	1183.004	1410.332	81.115	0,74	1915.31	1245.838	1498.412	90.302	0,78
LSTM-3	1875.060	1237.112	1466.105	116.096	0,78	1891.104	1209.955	1457.088	78.098	0,77	1915.31	1269.779	1537.356	80.927	0,80

When we analyze the results of K-Means parameter selection in Table 5.18, we see the DTW measure and decrease in cluster numbers produce better results as the amount of users increases.

Table 5.18: Results of K-Means Distance and Cluster Number Parameters (Call Time-Sum / Non-Discretized)

Users	100				1000				10000			
Cluster Numbers	3	5	7	15	3	5	7	15	3	5	7	15
K-Means (Euc.)	1095.284	1089.921	1076.103	1085.124	1176.234	1199.232	1189.245	1186.332	1204.343	1206.221	1211.764	1289.538
K-Means (DTW)	1074.213	1082.218	1085.916	1067.003	1168.217	1192.254	1150.107	1185.315	1196.646	1184.027	1201.285	1198.744

In DBSCAN models, number of outliers and clusters do not change strikingly. Table 5.19 presents the parameters.

Table 5.19: Parameters, Outliers and Cluster Numbers for DBSCAN (Call Time-Sum / Non-Discretized)

Users	Eps. and MinPts Parameters	Number of Outliers	Number of Clusters
100	0.65 / 3	30	7
1000	1.6 / 3	132	70
10000	1.3 / 3	333	295

Table 5.20 shows the number of predicted users in each user group for DBSCAN models. We see good amount of users in the most crowded cluster model for 10000

users, especially. The distributions of the users are presented in Figures 5.9, 5.10, 5.11, respectively. Red points in the figures denote outliers.

Table 5.20: Number of Predicted Users (Call Time-Sum/ Non-Discretized)

	Number of Predicted Users		
Models	100 users	1000 users	10000 users
LSTM-D-Gen.	70	868	9667
LSTM-D-Best	24	186	3174
LSTM-D-Best-5	64	496	6811

As the summary of all cluster based and DBSCAN based models, we notice improvements in RMSE/Exp.Value rates. Table 5.21 shows the results with all error metrics.

Table 5.21: Overall Results for Cluster Based Models (K-Means and DBSCAN) (Call Time-Sum / Non-Discretized)

USERS	100					1000					10000				
Error Metrics	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.
LSTM-C	1875.060	992.104	1067.003	84.961	0,56	1891.104	1052.277	1150.107	76.553	0,60	1915.31	1073.382	1184.027	78.254	0,61
LSTM-A	1875.060	1199.087	1262.735	104.668	0,67	1891.104	1066.600	1169.689	99.712	0,61	1915.31	1109.653	1217.350	102.572	0,63
LSTM-D-Gen.	1490.199	843.331	900.897	89.558	0,61	1677.572	745.65	807.147	76.300	0,48	1775.82	1121.327	1222.255	93.347	0,68
LSTM-D-Best	2432.301	1043.428	1156.768	74.406	0,48	1868.268	867.894	977.838	74.416	0,52	1997.81	761.572	897.118	64.379	0,44
LSTM-D-Best-5	1564.937	812.009	875.660	65.784	0,56	1927.744	548.288	590.380	55.651	0,30	1752.71	978.351	1064.350	73.218	0,60

We see the total improvements in Figure 5.12. LSTM-D-Best model predicts better in all user groups on average, with the exception of LSTM-D-Best-5 model predictions for 1000 users.

5.3.2 Discretized Time Interval Analysis

Table 5.22 shows the parameter settings for LSTM-1. As it is in Non-Discretized data set, wider LSTM cells and increase in epoch count get better results in the base LSTM model.

Table 5.22: Parameter Settings for Individual User-Based LSTM Models (Call Time-Sum/Discretized)

Sequence Length	Epoch	LSTM cells	Optimizer	Loss Function	Learning Rate	Batch Size	Activation Function
100	300	128	Adam	MSE	0.0001	32	Relu

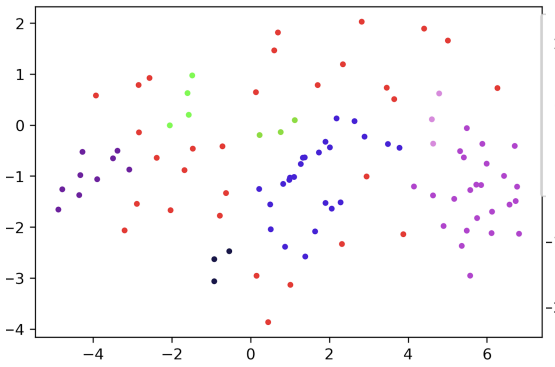


Figure 5.9: Clusters of 100 Users

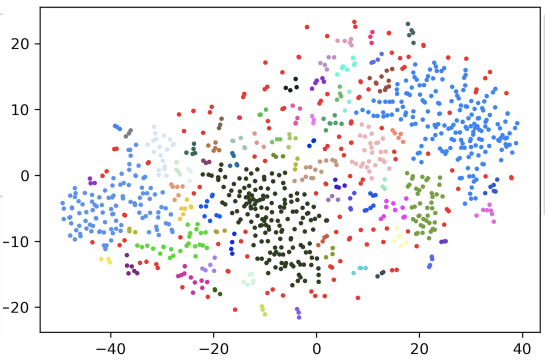


Figure 5.10: Clusters of 1000 Users

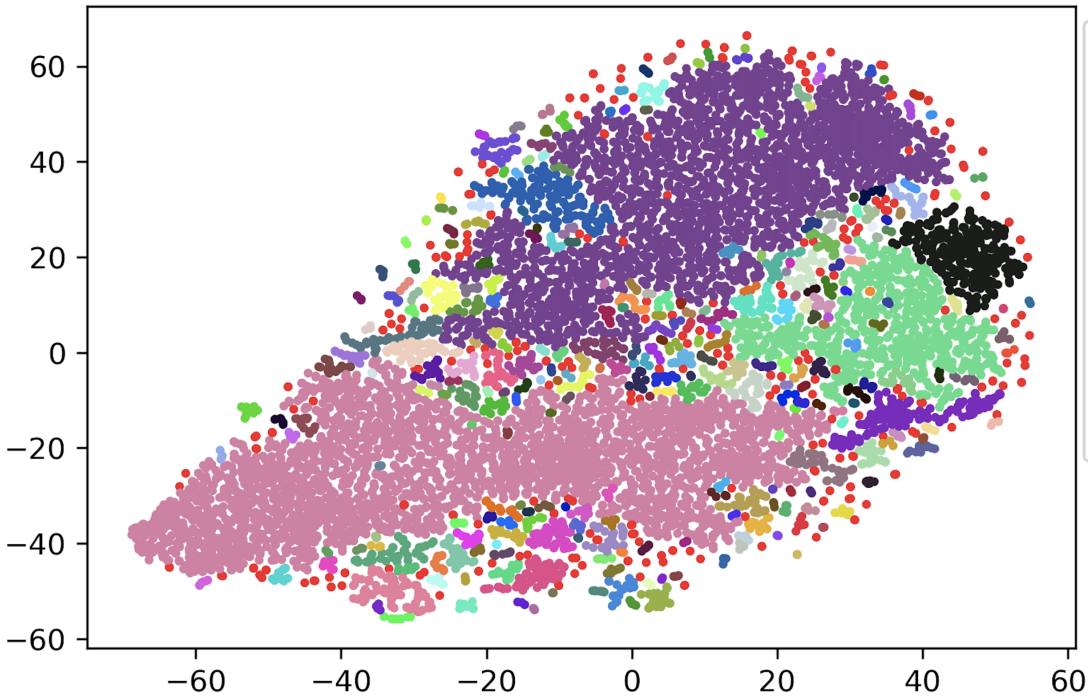


Figure 5.11: Clusters of 10000 Users with DBSCAN (Call Time-Sum/Non-Discretized)

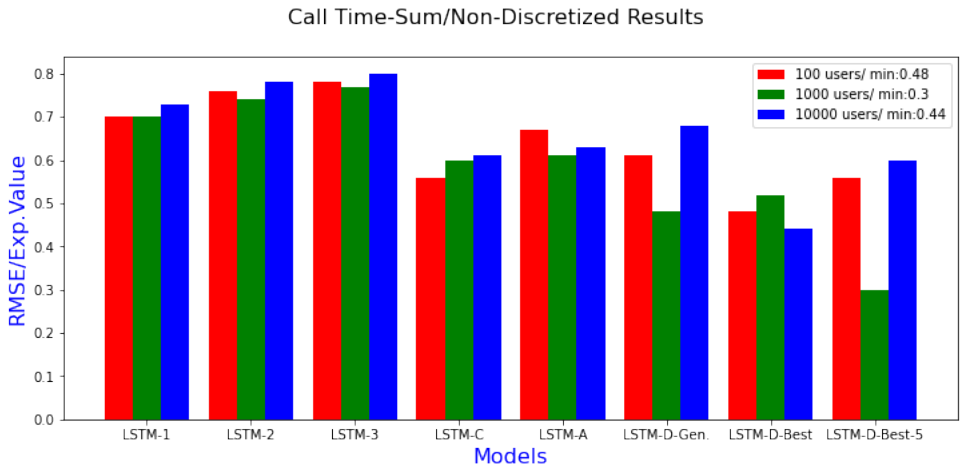


Figure 5.12: Overall Results of the Models (Call Time-Sum/Non-Discretized)

As the number of users increases in individual user-based models, there is improvement in decreasing the number of fully connected layers. LSTM-2 model gets the worst results, in which we do not see any positive effect of adding batch normalization and dropout layer. Table 5.23 shows the results for selecting the best individual model.

Table 5.23: Individual User-Based Model Results (Call Time-Sum/Discretized)

USERS	100					1000					10000				
Error Metrics	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.
LSTM-1	810.736	389.989	485.805	53.715	0.59	876.741	477.182	502.046	45.375	0.57	935.238	598.683	601.359	83.865	0.64
LSTM-2	810.736	397.183	493.398	54.246	0.60	876.741	493.866	602.483	60.294	0.68	935.238	492.479	612.248	57.627	0.65
LSTM-3	810.736	384.576	468.204	49.238	0.57	876.741	399.295	507.244	48.198	0.57	935.238	591.347	614.248	85.874	0.65

Table 5.24 shows the results for selecting the best parameters for cluster based model. DTW measure has the highest performance in all of the user groups with the increase in the number of clusters.

Table 5.24: Results of K-Means Distance and Cluster Number Parameters (Call Time-Sum/Discretized)

Users	100				1000				10000			
Cluster Numbers	3	5	7	15	3	5	7	15	3	5	7	15
K-Means (Euc.)	485.684	482.321	480.193	491.524	502.634	505.332	506.345	507.492	511.363	518.411	512.824	516.288
K-Means (DTW)	472.134	495.312	487.716	494.043	489.517	485.354	476.680	492.425	500.246	501.327	499.685	496.980

Table 5.25 presents the results for DBSCAN model parameters and number of clusters with outliers. When it is compared to Non-Discretized data set, we see that this data displays more scattered distribution.

Table 5.25: Parameters, Outliers and Cluster Numbers for DBSCAN (Call Time-Sum / Discretized)

Users	Eps. and MinPts Parameters	Number of Outliers	Number of Clusters
100	19.6 / 3	34	4
1000	1.1 / 3	173	97
10000	1.2 / 3	472	401

There are sufficient amount of users for the most crowded model of each user group, as it is seen in Table 5.26. Figures 5.13, 5.14 and 5.15 show the distributions of user groups with outliers in each figure.

Table 5.26: Number of Predicted Users (Call Time-Sum/Discretized)

	Number of Predicted Users		
Models	100 users	1000 users	10000 users
LSTM-D-Gen.	66	827	9528
LSTM-D-Best	56	245	4850
LSTM-D-Best-5	66	391	5562

In Table 5.27 of the results for cluster based and DBSCAN based models, the most remarkable prediction results are with LSTM-D-Best and LSTM-D-Best-5 models for 1000 and 10000 users. This implies that decreasing cluster numbers and separation of outliers cause good level of prediction. Figure 5.16 is the summary of all models used in discretized data set.

5.3.3 Section Discussion

For both of the problem sections, we see that LSTM-1 model has better results on average for individual user-based models. Neither using Batch Normalization and Dropout nor increasing the amount of fully connected layers at the end of base model

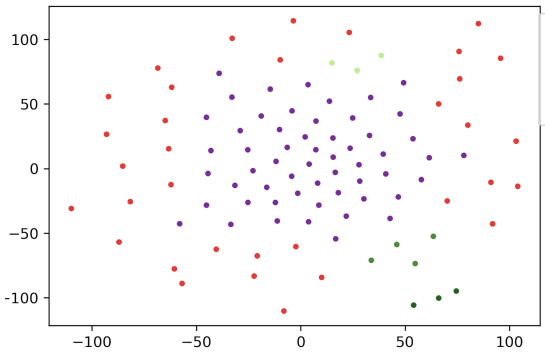


Figure 5.13: Clusters of 100 Users

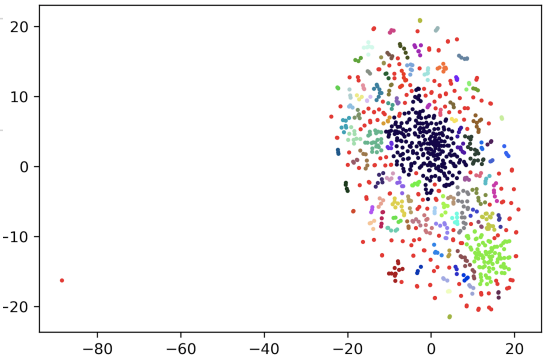


Figure 5.14: Clusters of 1000 Users

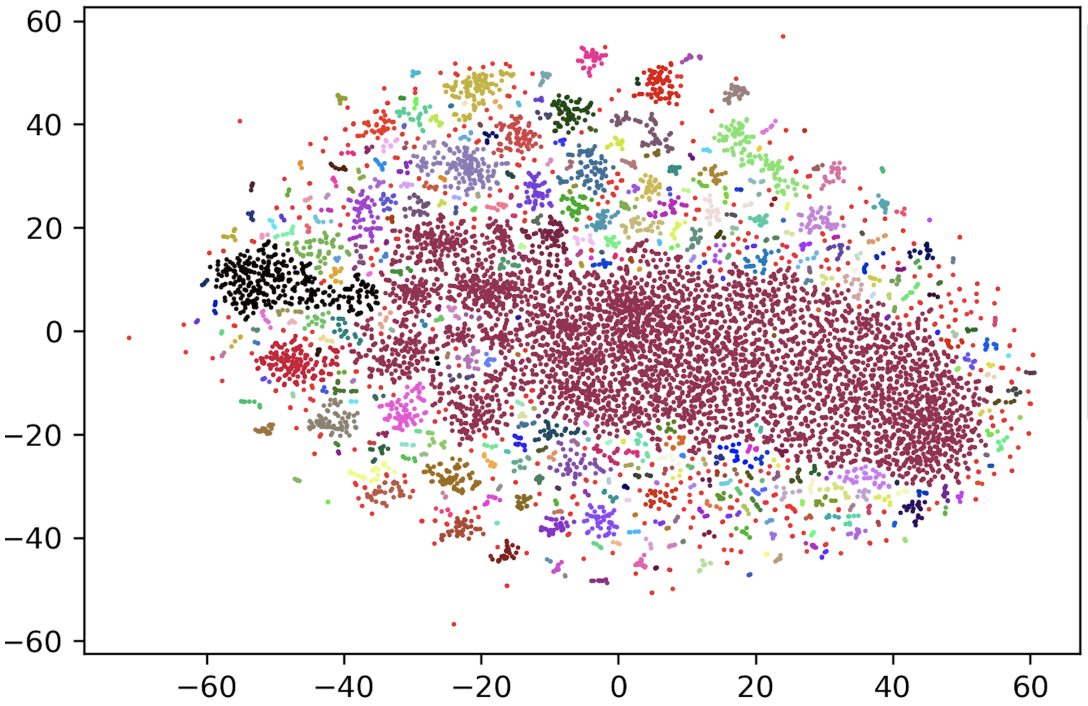


Figure 5.15: Clusters of 10000 Users with DBSCAN (Call Time-Sum / Discretized)

Table 5.27: Overall Results for Cluster Based Models(K-Means and DBSCAN) (Call Time-Sum / Discretized)

USERS	100					1000					10000				
Error Metrics	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.	Exp.Val.	MAE	RMSE	MAPE	RMSE/ Exp.Val.
LSTM-C	810.736	377.953	472.134	34.943	0.58	876.741	360.305	474.680	38.234	0.54	935.238	375.196	496.980	40.246	0.53
LSTM-A	810.736	362.885	445.505	28.358	0.54	876.741	372.335	475.405	35.326	0.54	935.238	421.384	519.849	43.642	0.55
LSTM-D-Gen.	820.128	305.642	439.977	27.284	0.53	806.549	335.526	411.414	31.582	0.50	829.702	365.473	476.552	42.645	0.57
LSTM-D-Best	829.146	307.758	435.534	24.832	0.52	829.571	188.963	270.169	27.971	0.32	807.207	166.244	245.725	34.392	0.30
LSTM-D-Best-5	826.148	351.353	451.979	36.765	0.54	758.987	254.276	316.085	30.802	0.41	775.025	282.024	349.577	38.348	0.45

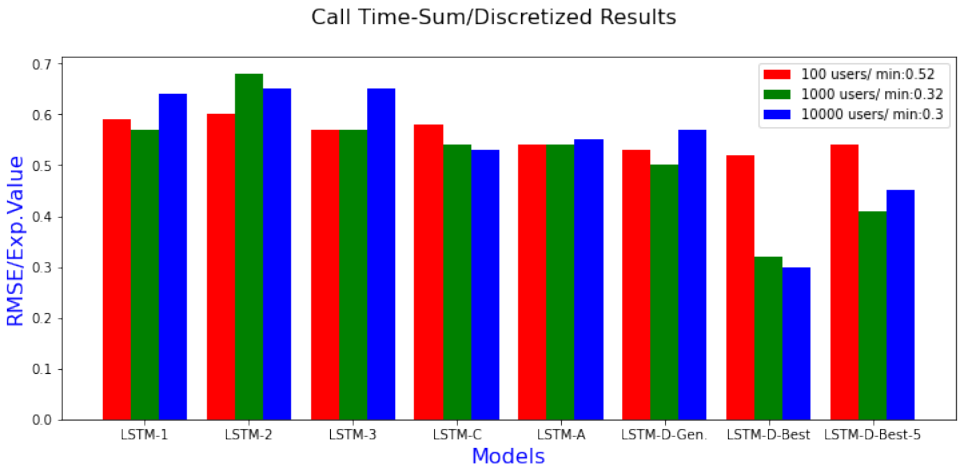


Figure 5.16: Overall Results of the Models (Call Time-Sum/Discretized)

have significant effects for the improvement of the models. When we compare discretized data with non-discretized data for the call time-sum problem, we find worse results in RMSE/Exp.Value rates at non-discretized data for all of the user groups. We interpret that discretization gets useful for the analyses in call time-sum problem.

Clustering with K-Means appears to be more effective in non-discretized data compared to discretized data. LSTM-C model, in which we use different numbers of cluster, gives best results in cluster analysis. DTW measure works better than Euclidean distance in cluster analysis.

In DBSCAN based models, one point to mention is the numbers of outliers. Although the outlier numbers is less than call count data, we see the positive effect of DBSCAN clusters especially in 1000 users data with looking to LSTM-D-Gen. model. Cluster numbers are close to each other in non-discretized and discretized data. In both of the data sets, we see that LSTM-D-Best model makes spectacular predictions. Different from call count data, in both of call time-sum data we see that LSTM-D-Best-5 model makes closer predictions to LSTM-D-Best model. The maximum improvement from the worst result of individual user-based models is %45 at DBSCAN models in non-discretized data for 10000 users, while it is %53,84 for discretized data. We again scale in data with DBSCAN methods as the amount of data increases and error rate decreases.

5.4 Analyses of Next Location Problem

As it is explained in Section 4.1.3, we find next locations in the trajectories of users with the help of connected base stations of outgoing calls. After labeling and clustering base station locations data, for each of the models mentioned below, we choose 20-day time intervals of users as training data. We test on the remaining 10-day time intervals. The models and testing procedures as the following:

- **The Most Frequent Place Model (MFP):** As a benchmark, we find the most visited locations of the user in the training data and compare it with the locations in the test data of the user. Overall accuracy is the average of each user's accuracy.
- **MLP Model (MLP):** Since the call numbers are not same on each day, we prepare the location matrix on the data of each user trimming to average call count for training data. Thus, the trajectories for each day in all users have the same lengths. Then, the location matrix is used for each user as training and test sets with fully connected layers on the model. The last layer has the number of nodes as the cluster numbers.
- **LSTM-2 Model (LSTM):** We prepare the data as it is for MLP model except that we use an embedding layer for the inputs before LSTM layers. We use LSTM-2 model mentioned in Subsection 4.2.1.2.

We add another approach to LSTM model for being more accurate on the location predictions. In LSTM-2nd-Prob. model, we try to find out the success rate with considering the location with the second highest probability on every prediction in the test set. Thus, we have two predictions at once and try to figure out if this increase contribute to accuracy results. In the LSTM-Current-Location, we change the second highest probability approach to consideration of current location for the second prediction. We present the overall results for each user group, number of clusters and models in Table 5.28.

In LSTM-Current-Location model, we realize that some predictions on current locations match with predicted next locations. Finding these matching locations becomes

Table 5.28: Next Location Prediction Accuracy Results for All Models

USERS	100			1000			10000		
Cluster Number	10	100	1000	10	100	1000	10	100	1000
MFP	63.921	33.843	33.126	59.431	34.067	32.026	58.921	34.572	32.469
MLP	65.209	51.023	43.530	63.971	49.057	41.283	59.034	37.322	35.751
LSTM	73.343	55.186	49.927	72.844	52.699	47.053	65.534	48.082	46.203
LSTM-2nd-Prob.	82.833	57.914	50.732	80.398	54.892	52.686	77.642	51.342	49.321
LSTM-Current-Location	88.997	77.466	74.040	85.511	75.391	73.309	83.312	71.216	69.246

signs for the stability of trajectories of users. The more matching numbers means the less users tend to move. Table 5.29 shows the results of matches and accuracy of considering only current locations in predictions.

Table 5.29: Accuracy Results and Number of Matching Predictions with Next Location and Current Location

Users	Avg. Length of Traj.	# of Clusters	Total # of Predictions	# of Matching Predictions	Accuracy(%)
100	7	10	7000	4328	82,14
		100		1827	74,26
		1000		865	61,35
1000	11	10	110000	62864	80,43
		100		21652	69,34
		1000		12532	65,46
10000	12	10	1200000	556347	79,46
		100		139424	65,48
		1000		98543	58,57

5.4.1 Section Discussion

The first remarkable result in the evaluation of the models is that both of the deep learning approaches, MLP and LSTM, beat the benchmark model in all of the user groups. Naturally, as the numbers of clusters increase, there is significant decrease in the accuracy results. That is because, we separate the locations into more discrete and standalone parts. Thus, predictions of the locations get more difficult . Base LSTM model, on the other hand, has specific amount of accuracy increase compared to MLP model. We can deduce that LSTM has the power of analyzing and predicting more

accurate in the sequential data.

We see that two-predictions-at-once approach results in increase in accuracy as a second conclusion. While LSTM-2nd-Prob. model has decrease in accuracy with as the cluster numbers increase, it does not prevent the small amount of superiority of the model compared to LSTM model. The best result is with the the model that considers current location, LSTM-Current-Location, in all of the user groups. For the purpose of analyzing scalability of the model, we see %33,33 increase in this model when compared to LSTM model for 10000 users and 1000 clusters.

Matching numbers in Table 5.29 show that selecting current locations as a first choice plays an important role for predictions. Especially in 10 clusters for each user group, current locations cause an amount of approximately %80 accuracy. As the cluster numbers increase, both of matching numbers and accuracy decrease. However, in the worst test group, 10000 users and 1000 clusters, we see that accuracy is above %55 in the %10 of the predictions which says that users in the randomly selected set are not moving to the distant places.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we define two main analyses for CDR data, regression and classification. In the regression problem, we test the models on two different data sets related to call counts and call time-sums of randomly selected users. In the classification task we try to predict next locations of the users in each user group.

Both **call count** and **call time-sum** problem use the same procedures in the analyses. We start the experiments with individual user-based LSTM models. Best individual model forms basis for cluster based LSTM models. Then, we use the selected LSTM model on the outlier separating models. To our knowledge, this approach is the first work that uses outlier separation in the field of CDR data analysis. We use all of the models in non-discretized and discretized data. Discretization of the time intervals ensures stable results for the same problem while experimenting on two different data sets makes all of the proposed models applicable to the same kind of problems. After the experiments, we see an increase of %61,20 in non-discretized data and %66,28 in discretized data for call count problem from worst individual user-based LSTM model to outlier separating LSTM model while this rate is %45 on non-discretized and %53,84 on discretized data for call time-sum problem.

As the classification task, we predict **next locations** of the trajectories of the users via deep learning approaches. We compose a location matrix for each user and from training values (20-day data of the user), we predict the next 10-day data. The first experiments show the superiority of deep learning approaches (MLP and LSTM) to the benchmark most frequent place model. We expand the prediction strategy to **two-predictions-at-once** and use second probable or current location as the second choice in the predictions. These optimized models show dominance to the base models and

yield promising results. There is an increase amount of %33,33 from base LSTM model to the best of latter strategy models, LSTM-Current-Location, in the accuracy of the results.

To be able to show the scalability of the models, we experiment on 100, 1000 and 10000 users. For each of the problems, we have the same kind of improvements in the group of 10000 users. Thus, these experiments show that the proposed models are scalable in CDR data analyses.

The future directions gathered from this work are listed as:

- We apply the models on limited numbers of users. With the computing power increase in the experiment environment, the models can be tried on more or all of the users in the data set.
- Changing the CDR data set can be beneficial in validating of the models.
- The models can be applied to different problem sets related to CDR data, like predicting on date or time of action.
- The calculations are taking long times for especially 10000 users group. Thus, high performance computing approaches can be beneficial for speeding up the calculations.

REFERENCES

[1] M. Deshpande, "Perceptrons: The first neural networks." https://pythonmachinelearning.pro/perceptrons-the-first-neural-networks/, 2017. [Online; accessed 13-March-2020].

[2] H. Ramchoun, M. Amine, M. A. Janati Idrissi, Y. Ghanou, and M. Ettaouil, "Multilayer perceptron: Architecture optimization and training," *International Journal of Interactive Multimedia and Artificial Inteligence*, vol. 4, pp. 26–30, 01 2016.

[3] A. Jaffe, "Lstm rnn for classification of ahe," Master's thesis, 01 2017.

[4] T. Lillicrap and A. Santoro, "Backpropagation through time and the brain," *Current Opinion in Neurobiology*, vol. 55, pp. 82–89, 04 2019.

[5] C. Olah, "Understanding lstm networks." https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. [Online; accessed 15-March-2020].

[6] Y. Chen, B. Hu, E. Keogh, and G. Batista, "Dtw-d: time series semi-supervised learning from a single example," pp. 383–391, 08 2013.

[7] C. Cassisi, P. Montalto, M. Aliotta, A. Cannata, and A. Pulvirenti, *Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining*. 09 2012.

[8] S. Elagib, A. Hashim, and R. Olanrewaju, "Cdr analysis using big data technology," pp. 467–471, 09 2015.

[9] S. Ho, M. Xieb, and T. Gohb, "A comparative study of neural network and box-jenkins arima modeling in time series prediction," *ACM-Computers and Industrial Engineering,Volume 42, Issue 2-4*, 2002.

[10] J.-H. Wang and J.-Y. Leu, "Stock market trend prediction using arima-based neural networks," *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 4, pp. 2160–2165, 1996.

[11] H. Cagdas, C. Aladag, E. Egrioglu, and C. Kadilar, "Forecasting nonlinear time series with a hybrid methodology," *Applied Mathematics Letters*, vol. 22, pp. 1467–1470, 09. 2009.

[12] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211.

[13] I. Khandelwal, R. Adhikari, and G. Verma, "Time series forecasting using hybrid arima and ann models based on dwt decomposition," *Procedia Computer Science*, vol. 48, 12. 2014.

[14] P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, 01. 2003.

[15] S.-L. YU and Z. Li, "Stock price prediction based on arima-rnn combined model," *DEStech Transactions on Social Science, Education and Human Science*, 03. 2018.

[16] M. Rhanoui, S. Yousfi, M. Mikram, and H. Merizak, "Forecasting financial budget time series: Arima random walk vs lstm neural network," *IAES International Journal of Artificial Intelligence*, vol. 8, pp. 317–327, 12. 2019.

[17] M. Ashour and R. Helmi, "Improving time series' forecast errors by using recurrent neural networks," *ICSCA 2018: Proceedings of the 2018 7th International Conference on Software and Computer Applications*, pp. 229–232, 02. 2018.

[18] R. Madan and P. S. Mangipudi, "Predicting computer network traffic: A time series forecasting approach using dwt, arima and rnn," *2018 Eleventh International Conference on Contemporary Computing (IC3), Noida*, pp. 1–5, 2018.

[19] A. Temür, M. Akgün, and G. Temür, "Predicting housing sales in turkey using arima, lstm and hybrid models," *Journal of Business Economics and Management*, vol. 20, pp. 920–938, 07. 2019.

[20] S. Siami Namini, N. Tavakoli, and A. Siami Namin, "A comparison of arima and lstm in forecasting time series," *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL*, pp. 1394–1401, 12. 2018.

[21] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *https://arxiv.org/abs/1909.00590*, 09 2019.

[22] Y. Leng, L. Rudolph, A. Pentland, J. Zhao, and H. Koutsopoulos, "Managing travel demand: Location recommendation for system efficiency based on mobile phone data," *Proceedings of Data for Good Exchange*, 2016.

[23] N. C. Chen, W. Xie, R. E. Welsch, K. Larson, and J. Xie, "Comprehensive predictions of tourists' next visit location based on call detail records using machine learning and deep learning methods," *2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 1–6, 2017.

[24] M. S. Mahdizadeh and B. Bahrak, "A regression framework for predicting user's next location using call detail records," *arXiv:1912.10438*, 2019.

[25] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "Deepmove: Predicting human mobility with attentional recurrent networks," *the 2018 World Wide Web Conference*, pp. 1459–1468, 2018.

[26] Y. Leng, "Urban computing using call detail records:mobility pattern mining, next-location prediction and location recommendation," Master's thesis, Department of Civil and Environmental Engineering, Department of Electrical Engineering and Computer Science,Massachusetts Institute Of Technology - USA, 2016.

[27] A. Crivellari and E. Beinat, "Lstm-based deep learning model for predicting individual mobility traces of short-term foreign tourists," *Sustainability*, vol. 12, p. 349, Jan. 2020.

[28] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using lstm networks," *2018 IEEE 29th Annual International Symposium on Per-*

sonal, Indoor and Mobile Radio Communications (PIMRC), pp. 1827–1832, Sep. 2018.

[29] C. Huang, C. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, Oct. 2017.

[30] X. Shao, D. Ma, Y. Liu, and Q. Yin, "Short-term forecast of stock price of multi-branch lstm based on k-means," pp. 1546–1551, 11 2017.

[31] S. Vadyala, N. Betgeri, E. Sherer, and A. Amritphale, "Prediction of the number of covid-19 confirmed cases based on k-means-lstm," 06 2020.

[32] J. Zhang, F. Chen, and Q. Shen, "Cluster-based lstm network for short-term passenger flow forecasting in urban rail transit," *IEEE Access*, vol. 7, pp. 147653 – 147671, 09 2019.

[33] F. F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[34] K.-L. Du and M. Swamy, *Neural Networks and Statistical Learning*. 10. 2013.

[35] B. Kıvılcım, "Modeling brain networks with artificial neural networks," Master's thesis, Department of Computer Engineering, Middle East Technical University - TUR, 2019.

[36] R. Asadi and S. Abdul Kareem, "Review of feed forward neural network classification preprocessing techniques," *AIP Conference Proceedings*, vol. 56718411146, 06 2014.

[37] H. Ullah and M. Bhuiyan, "Performance evaluation of feed forward neural network for image classification," *Journal of Science and Technology*, vol. 10, 05 2018.

[38] H. Gholamhosseini, D. Luo, and K. Reynolds, "The comparison of different feed forward neural network architectures for ecg signal diagnosis," *Medical engineering physics*, vol. 28, pp. 372–8, 06 2006.

[39] S.-i. Mayekawa, "An efficient algorithm for feed-forward neural network regression analysis," *Behaviormetrika*, vol. 22, pp. 67–90, 01 1995.

[40] V. Dave and K. Dutta, "Application of feed-forward neural network in estimation of software effort," 03 2020.

[41] D. Rios and P. Muller, "Feedforward neural networks for nonparametric regression," 03 1998.

[42] L. Tawfiq and A. Adnan Thirthar, "Improving gradient descent method for training feed forward neural networks," 03. 2019.

[43] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," vol. 2, pp. 1045–1048, 01 2010.

[44] S. Liu, L. Bai, Y. Hu, and H. Wang, "Image captioning based on deep neural networks," *MATEC Web of Conferences*, vol. 232, p. 01052, 01 2018.

[45] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 4, 09 2014.

[46] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," *Complex Systems*, vol. 3, 1989.

[47] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, pp. 157–166, March 1994.

[48] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[49] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. Aljaaf, *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science*, pp. 3–21. 01 2020.

[50] A. Saxena, M. Prasad, A. Gupta, N. Bharill, o. Patel, A. Tiwari, M. Er, and C.-T. Lin, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, 07 2017.

[51] D. Chang, Y. Ma, and X. Ding, "Time series clustering based on singularity," *International Journal of Computers Communications Control*, vol. 12, p. 790, 12 2017.

[52] P. Berkhin, "Survey of clustering data mining techniques," 2002.

[53] J. A. Hartigan, *Clustering Algorithms*. USA: John Wiley Sons, Inc., 99th ed., 1975.

[54] I. Keleş, "Methods for location prediction of mobile phone users," Master's thesis, 07 2014.

[55] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *PVLDB*, vol. 1, pp. 1542–1552, 08 2008.

[56] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, pp. 70–80, 01 2004.

[57] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, p. 226–231, 1996.

[58] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines vinod nair," *Proceedings of ICML*, vol. 27, pp. 807–814, 06 2010.

[59] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[60] I. Sergey and S. Christian, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[61] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[62] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," 01 2018.

[63] R. Manuca and R. Savit, “Stationarity and nonstationarity in time series analysis,” *Physica D: Nonlinear Phenomena*, vol. 99, pp. 134–161, 12 1996.

[64] S. E. Said and D. A. Dickey, “Testing for unit roots in autoregressive-moving average models of unknown order,” *Biometrika*, vol. 71, pp. 599–607, 12 1984.