Santa Clara University

Information Systems and Analytics Department

Date: Oct 20, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Muhammad Adeel Sultan Khan

ENTITLED

Uber Project

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MANAGEMENT INFORMATION SYSTEMS

Thesis Advisor Dr. Name

Thesis Reader Dr. Name

Chairman of Department
Dr. Name

Thesis Reader Dr. Name

Thesis Reader Dr. Name

Thesis Reader Dr. Name

Uber Project

By

Muhammad Adeel Sultan Khan

Master's Thesis

Submitted in Partial Fulfillment of the Requirements
for the Degree of Master's
in Management Information Systems
in the Leavey School of Business
Santa Clara University, 2018

Santa Clara, California

Acknowledgments

I would like to express my sincerest appreciation and gratitude to my advisor Dr.Yasin Ceran for his continued support and guidance throughout this thesis research project. Dr.Ceran's expertise and skills in the research and data analytics areas have been very supportive for me throughout this project. I have been able to attain invaluable skills related to data science, machine learning and optimization of complex problems through the research work done in this project. Also, the associated thesis writing really helped me in improving my writing and comprehension skills

I am highly thankful to God, my parents and my brothers for their continued support, guidance and empowerment throughout my studies. In addition i am also thankful to Ms Ramie Fernandez, Ms Minh Virasak and Professor Vasu Kadambi for their help and support during my grudate studies at SCU. I feel confident and optimistic to utilize the skills and the experience gained in this project throughout my professional career. With these skills in hand, i look forward to pursue a successful career in data science, statistical modelling and machine learning. Having these skills, I also aspire to attain inimitable and sustainable competitive advantages for all the stakeholders that i would work with and to contribute positively towards making our world a better place for everyone

Dedications

This thesis is dedicated to my beloved father (late) Javed Sultan Khan for his eternal love and motivation. His virtues of honesty, dedication, perseverance and faith have been the guiding forces throughout this thesis project and will be my principles for all of my next endeavors in life

Department of Information Systems and Analytics
Santa Clara University
Santa Clara, California
2019

ABSTRACT

This research thesis is focused on developing optimal allocation of Uber drivers to the most profitable and economically feasible geohash locations in a city. In doing that, we aspire to provide a win-win solution for all the stakeholders including passenger, driver as well as the ride sharing company like Uber,Lyft etc. We start by dividing the city into equally spaced clusters and then calculating the distance between drivers, passengers and final destination of passengers. Subsequently, we calculate expected profits for drivers considering their current location, location of customer request and the travel destination of passenger. Through this optimal allocation model, we propose to reduce driver's idle cruising time and the waiting time of passenger after requesting a ride. Thus, the current imbalance between supply and demand of Uber drivers is expected to be minimized

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

As world's population is increasing rapidly, dynamic and robust infrastructure as well as technologically advanced, yet reliable, and efficient transportation solutions need to be in place to facilitate urban passenger traffic movement. This also requires diligent urban metropolitan planning to ensure that the increased traffic congestion caused by ride hailing companies like Uber and Lyft is well adjusted by meticulous road network planning. In the recent times, technology has revolutionized every aspect of our decision making and is continuously providing better and more productive ways to fulfill respective demands from both the user and service provider's perspective. However, with increasing competition and more available customer choices to select a particular service, the successful and long-lasting companies would be those investing in research and devising more innovative ways to conduct business and thereby able to optimize their revenues. Similarly, internet based technology is disrupting transportation services and thus providing more smart choices for customers and offering various utility options in terms of easy to use and ubiquitous mobile applications

Mobile software applications are disrupting the way clients get connected to service providers and transforming the way that matches the supply of business providers with customer demand. Ride hailing services is one great example of this technological disruption and the way passengers request rides has been continuously evolving. Since retaining existing customers and increasing market share are crucial success factors

for ride hailing companies, therefore, those who design their software in a way that it reduces the imbalance between supply and demand of drivers would take the lead in the thriving competition. This would also ensure that passengers get the best possible ride hailing service in terms of both time and money spent on the ride. Nevertheless, these companies would have to ensure that a passenger requesting a ride has the minimum waiting time at the lowest possible price, and also the available drivers have the minimum idle travelling time before they get a ride request. One of the foremost goals of ride hailing companies is to provide their drivers with the highest possible expected profit which is higher than the street hailing conventional taxi drivers income. This would be the win-win situation for all the stakeholders and that is the core research area of this thesis where we want to increase the total social benefit for all the stakeholders by dynamically allocating drivers in the most efficient manner. We want to achieve the mentioned objective by also considering the factors such as driving time of driver, waiting time of passenger and the expected profit for the driver in serving the passenger's ride hailing request besides also focusing on which drivers to allocate to which demand hotspots at different time periods of a day

This thesis research is expected to be based on detailed data analytics and machine learning tools in order to analyze historical data and make demand predictions for expected upcoming passenger ride requests across different locations in a particular city. Through this, we want to be able to guide drivers to reach the potential high demand areas and facilitate ride sharing requests besides adjusting price based on peak demand time periods. This would eventually enable the companies like Uber to narrow down the supply and demand gap using effective and efficient demand prediction systems based on mathematical models like ARIMA and Poisson distribution. This would enable us to accurately forecast customer demand and thereby adjust the supply of drivers to match the impending passenger demand so that passengers and drivers could get their

interests served in the best possible way. Providing ride-hailing services in almost every major city of the world, Uber is one of the most successful technology-based ride hailing companies and was estimated to be valued at $70 billion in May 2018. Having customer base of around 75 Million users, Uber went for the initial public offering of its stocks on May 10,2019 and was worth almost $83 billion considering the robust performance and ever-increasing customer base of the company. Moreover, Uber has completed 10 billion trips globally and has been operating in over 80 countries and 700 cities in a period of just 9 years since its initial launch of application software in 2009

In our proposed research, we aspire to collect real-time spatiotemporal data of 41 Uber drivers who provided ride hailing services in San Francisco. The data would be in a form of panel dataset and would have various columns including the variables such as driver ID, time and location of pick-up and drop-off and cost of ride for a passenger as well as geohash locations. This research project is thus expected to be an important contributor towards solving one of the major constraints that ride hailing companies face everyday during their operations. It is an ambitious and challenging endeavor towards optimizing real life scenario by analyzing the available Uber passenger demand data and identifying existing patterns, trends and relationships to streamline the passenger and vehicle availability. Therefore, this would be a great effort towards reducing idle vehicle time and to provide more convenience to users by reducing vehicle wait time at their points of pick-up. The trend in the rise and growth of ride hailing companies in the recent years can be analyzed from a line chart which is shown in the following figure as follows:

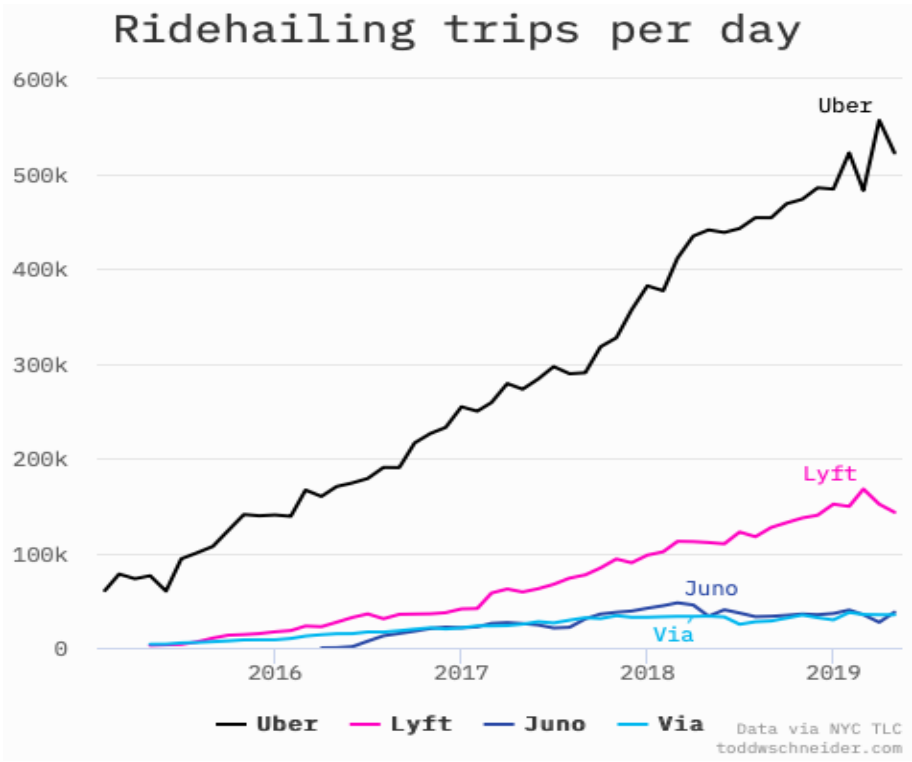

Fig. 1.1: New York City Daily Trips (Schneider (2019))

The above figure 1.1 shows the continuous increasing trend in the daily ride hailing trips in New York city and also shows that Uber is the majority market share holder. With over 500,000 daily trips, Uber definitely requires robust route optimization models to reduce supply and demand imbalance in ride sharing services. This research requires extensive effort and continuous focus towards identifying optimal driver allocation strategies for routing optimization problems. The research would be using Machine Learning and Artificial Intelligence tools and techniques besides using available statistical computational packages like Python and R which would be utilized to analyze the available dataset of 41 Uber drivers. Since our data is in univariate time series format where we have only one set of observation or variable measured over time, the most appropriate Machine Learning models that we plan to implement include Seasonal Auto Regressive Integrated Moving Average (SARIMA) which predicts and forecasts fu-

ture demand data points and is suitable for the data which has seasonality, periodicity, pattern and non-stationarity

The data is non-stationary because the mean and variance are not constant and change over a period of time due to fluctuating demand patterns based on different time periods. The purpose of this research is therefore to be able to use the spatiotemporal distribution of Uber passenger time-series data and to make predictions for future expected passenger demand for Uber ride hailing services using the machine learning models mentioned above like Poisson distribution and SARIMA. This would help in identifying useful patterns,trends and relationships with ample opportunities for companies like Uber and Lyft to make better business strategies and hence increase profits and their relevant market share. However, special emphasis need to be placed in ensuring that in all technological functioning of their business; the core ethical principles of social and federal regulations are complied with appropriately. This involves secrecy and privacy of available customer data as well as their preferences and security concerns while using the ride hailing services

One of the most important Machine Learning algorithms that we aspire to use in this research includes Auto Regressive Integrated Moving Average (ARIMA) which would be used to compute the demand predictions for Uber services in a particular location and at a certain time period. Based on these forecasted demand predictions, we would allocate drivers to their most profitable and economically feasible next passenger location. In doing this driver allocation, we want to incorporate the constraint that the total number of customer requests for ride services at a particular time period and in a certain geohash location is always greater than or less than the total number of drivers allocated there. This would be one of our constraints in the mentioned driver and Uber optimization problems that we look forward to solve. ARIMA is a very robust and

useful statistical model used for time series forecasting and it is an ideal algorithm for our use case where we want to forecast the demand predictions of passengers for Uber ride sharing services

ARIMA also has the potential to incorporate the fluctuations and periodic patterns in passenger demand and also takes care of seasonality effects in the time series demand data. Since we would be allocating drivers based on the predictions and forecasts generated by ARIMA model, this demand forecasting is very crucial in our research and is expected to play a pivotal role in our mission to reduce the supply and demand imbalance between passengers and drivers. Secondly, we would calculate the expected number of passenger requests emerging from different locations by also incorporating the periodicity in demand for Uber services including daily, weekly and seasonal fluctuations. Subsequently, we would also calculate expected profit for a driver in facilitating a customer's ride request and based on that we would optimally allocate drivers to the most profitable geohash location so that all the stakeholders including driver, passenger and the company like Uber are satisfied to the maximum extent. To study the above mentioned effects of daily, weekly, monthly and holiday periodicity patterns with reference to passenger demand, we would use another very useful statistical model called Poisson distribution to analyze the effects of changing passenger demand for Uber services over different time periods

Poisson distribution is a discrete probability distribution method used for analyzing univariate time series data where each event occurring is independent of the time since the last event. It is also very useful for instances where we want to model a discrete random variable X which in our case denotes the number of passenger requests for Uber in a certain time period and in a certain cluster/location. Also, to incorporate the seasonality effects on passenger demand where seasonal inconsistent demand due

to large crowd events like games, concerts, etc cause high fluctuations in demand, we would use Weighted Time varying Poisson Model, which assigns higher weights to most recent time periods and use exponential smoothing approach to calculate weights. The historical data of 41 Uber drivers is the starting point for this purpose of identifying the seasonalities, trends and patterns by performing in-depth data analytics in R. The results and findings of this data mining exercise are attached and discussed in chapter 4, section 4.2 and 4.3. We also want to be able to increase the effect of most recent events in our algorithm and that can be achieved by assigning higher weights to them as compared to the weights assigned to older events

Several factors encouraged us to choose this challenging thesis topic and the reasoning behind it is explained as follows. As technology is increasingly becoming ubiquitous and penetrating in every domain of our daily living and has almost become an absolute necessity, transportation recommendation systems such as Uber are becoming increasing popular with the technological advances and with the growing use of cell phones for almost every daily activity. Also, it is expected that the global ride hailing services industry would continue to grow to an estimated $285 billion by 2030 (MarketWatch (2017)), signifying the importance of such technology based transportation solutions. Therefore, through this research we aspire to reduce passenger waiting time, increase capacity utilization and throughput besides increasing the total social benefit to all the stakeholders

We also intend to recommend profitable locations to drivers using optimal route allocation which would help increase their revenues, reduce traffic jams and also minimize fuel consumption by helping drivers avoid random cruising strategies in search of passengers. This would enable drivers to increase their number of trips and help passengers in saving on travelling time in the most cost effective and efficient way

We also aim to provide drivers the ability of scheduling flexibility so that they can choose which passenger locations they should go to next based on total expected profit and thus strive towards maximizing their expected revenues during their available times by targeting the high demand regions.This would also be facilitated by the availability of heat maps and historical passenger-driver movement patterns in terms of graphs denoting daily, weekly, and monthly periodicity in different locations of a city. Certain events like sports and major league tournaments result in sharp increase in the demand for ride hailing services and also cause major traffic jams around the stadium locations as people strive to reach the venue on time. Therefore, we want the drivers to be aware of such events happening and thus take appropriate steps to ensure effective and efficient ride availability to the passengers

Also, events like summer vacations and major holidays like Thanksgiving, Christmas and Labour day events also affect the Uber services in a dynamic way. In general, we want to ensure that the supply of Uber driver services is allocated in accordance with the travel intensity of each location within a city as we are dividing each city into clusters/geohashes and thus manage each geohash with respect to its demand in different time periods of a day. Therefore, we aim to provide this information to the drivers by using data analytics on historical passenger and driver movement patterns and the time periods of such events happening so that optimal driver allocation can be achieved and passengers get the best available ride sharing services. Economy, environment and daily movement of people could be benefited vastly by efficient routing of vehicles including taxis and ride sharing cars. Therefore, research in this area is the absolute requirement of current times and would be required in the future as well to make our routine daily travels as efficient, smooth and flexible as possible

The main focus of our research is therefore as follows:

To reduce supply and demand imbalance and improve drivers' mobility intelligence by accurately predicting future forecast of Uber passenger service demand for a certain time period of t=30 minutes across different geohashes and providing that information to drivers for effective fulfillment of respective customer demand

Chapter 2

Related Work

Analyzing historical passenger and driver movement patterns and predicting urban human mobility in the next time period are getting unprecedented attention by data scientists and optimization research specialists. Numerous efforts have been devoted towards using the data generated from sensors and Global Positioning Systems (GPS) devices to improve transportation infrastructure and reduce road network congestion by utilizing the urban mobility patterns to its optimal level. GPS location devices and the use of associated location monitoring software like Google Maps have enabled researchers and companies to gather massive volume of data, perform extensive data mining on it and thereby help top management in making better business strategies and take smart decisions by understanding historical trends, patterns and relationships in that data. Hundreds of research papers have been written and countless efforts are done using the same topic as ours. The following are some of the research papers that are selected for cross reference and therefore read in order to better understand the driver allocation problem

With advances in computing technology and the availability of robust statistical and machine learning models, the historical data can be utilized to train the algorithms and then subsequently assist in making better and more refined predictions for the end users. Li et al. (2012) proposes ARIMA based models to predict human mobility patterns in an urban taxi transportation system by identifying trends and using historical

GPS data of the number of passenger pick-ups quantity (PUQ) from urban hotspots. This research was done to forecast the number of passengers available in a particular urban hotspot at a certain point in time and claimed that the findings could be useful for drivers, passengers, traffic police and urban planning staff. Li et al. (2012) uses naive method, Bayesian networks and auto-regressive integrated moving average (ARIMA) model to forecast the predicted future values of number of passengers expected to be in a certain geographical hotspot by using the observed historical data available in the dataset. Their research is facilitated by the available GPS trajectories of past spatio-temporal passenger pick-up and drop-off variations in a particular hotspot

Chang et al. (2010) uses data mining and clustering algorithm on the historical data to predict taxi demand requests and the related supply of taxis with respect to changing time, weather and location. This research proposes better management of taxi fleets and fulfillment of customer demand to the best potential by using context-aware demand prediction of geographical hotspots in a major urban metropolitan city. Three different clustering algorithms are used including k-means, agglomerative hierarchical clustering and DBSCAN for the location and request history datasets of taxis and customer requests for them. K-means is a popular unsupervised machine learning clustering algorithm for vector quantization and for cluster analysis in data mining. k-means clustering divides n observations into k clusters in which each observation is assigned to the nearest cluster with the nearest mean by using straight line distance measure such as Euclidean distance. Agglomerative Hierarchical clustering is a bottom up approach in which each observation starts in its own cluster, and then pairs of clusters are merged as one moves up the hierarchy. The resulting figure is a dendrogram which is a tree like structure representing the arrangement of clusters

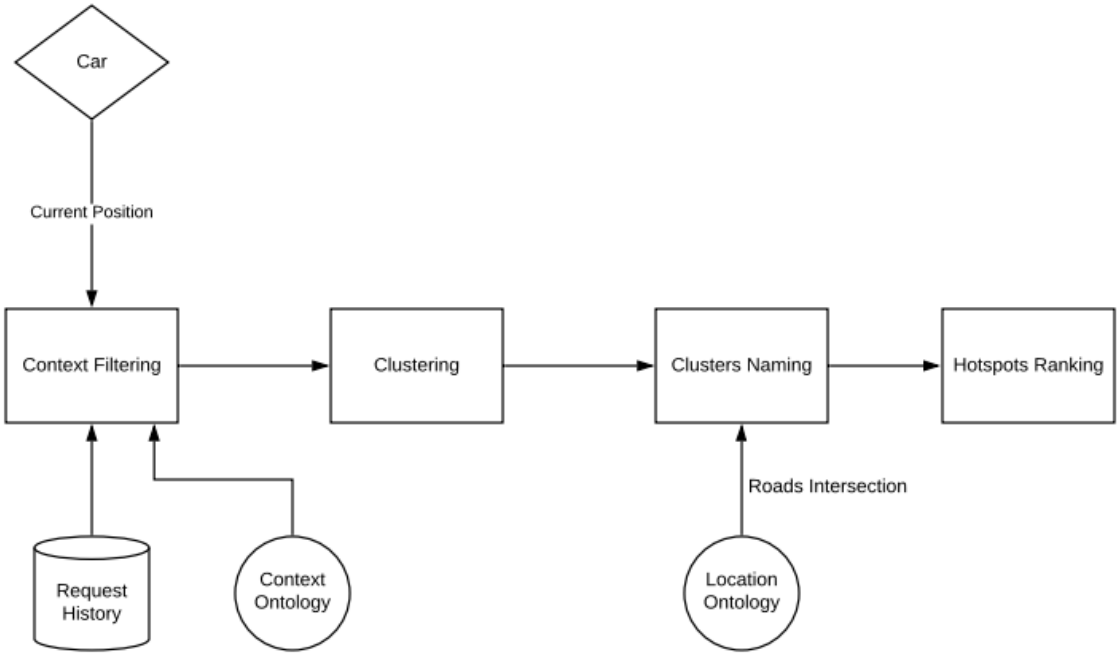

Fig. 2.1: Taxi demand prediction architecture

The main idea in figure 2.1 is presented by Chang et al. (2010) in their research paper as described earlier, it shows the methodology to be used in guiding a driver to the next passenger potential demand location once the driver drops off current passenger to the destination. The architecture presents the technique to be used in predicting the potential demand hotspots and presenting the results to the driver in an easy to use navigational pane. The steps include mining historical passenger demand requests database and then using clustering algorithm to identify and form clusters to detect locations of respective passenger demand. Subsequently, the existing distance between the driver and the clusters are calculated and the relevant hotness scores are tagged with those clusters to emphasize the potential benefit for the driver in case the driver chooses that particular location as a next passenger pick up point. This is intended

to be helpful for drivers in instantly choosing their next passenger location rather than opting for random cruising strategy in search of new passengers, thereby saving on fuel costs as well as reducing their idle driving time

Yuan et al. (2013) research leverages the intelligence of experienced drivers from a large pool of past taxi trajectories in choosing the best possible route for the efficient and effective usage of concerned resources by means of sensor and GPS equipped taxis. This data eventually is transmitted through cloud computing software and assists current drivers with the fastest and optimal route to a given destination and thus results in a win-win situation for both customer and taxi driver. This research focuses on performing real time data analytics on the historical data and then transforming the results to match drivers expectations and finally using clod technology to present the findings to drivers

Deng and Ji (2011) uses spatio-temporal structure of taxi services in Shanghai, China using exploratory spatial data analysis (ESDA) to identify spatial structure of taxi services over a period of specified time with respect to business and social events and routine daily activities. This research focuses particularly on identifying seasonality effects, demand upsurge patterns and the respective locations of such changes in passenger demand

Tang et al. (2013) focuses on providing taxi drivers the most cost effective and efficient way to pick-up next passenger using the available GPS data of taxicabs and modelling the problem of finding a profit maximizing passenger using a Markov Decision Process (MDP). Having been used extensively for optimization problems and recently becoming popular in the reinforcement learning aspect of machine learning, Markov decision process is a discrete time mathematical modelling framework which helps in choosing the most optimal output from a large number of random and all possible uncontrolled events

Moreira-Matias et al. (2012) uses time series forecasting techniques to predict the spatial-temporal distribution of passenger demand in 63 taxi stands in the city of Porto, Portugal and using the forecasts to improve and strengthen the driver mobility intelligence. Their research is aimed at helping taxi drivers in choosing the best taxi stand to go to after a passenger drop-off by building a mathematical model to predict the number of taxi services expected to emerge at a certain stand for a time interval of P = 30 minutes. Moreira-Matias et al. (2012) uses Poisson time varying model, Auto regressive integrated moving average (ARIMA) and sliding window Ensemble frameworks to calculate forecasts for the expected services to emerge at a certain spatial location. The results indicate that sliding window Ensemble method performed better than other time series forecasting techniques with sMAPE accuracy error of 76% in correctly forecasting taxi services in both space and time using an aggregation period of 30 minutes

Zhao et al. (2015) proposes a method of trajectory clustering based on decision graph and data field in order to identify potential taxi hotspots and cluster centers in Wuhan city, China by analyzing taxi trajectory data with respect to holiday, weekend and a weekday. Ihler et al. (2006) analyses seasonality, trend and noise in time series data and uses specific contexts of web access logging, freeway traffic monitoring and building security logs. This type of data exhibits daily, weekly and monthly periodicity as well as non-homogeneity based on unexpected and unusual events such as traffic accidents or a super bowl game at a local stadium leading to a significant increase in vehicles on highways and roads. This paper proposes an unsupervised learning technique using time-varying Poisson process to identify anomalous events in the historical data and uses that insight to predict similar future events. This information can then be used to estimate an event popularity and attendance or to predict the extent of traffic jam on a certain highway of a particular city

Many big companies are utilizing the power of Hadoop open source software framework for storing massive quantities of data on a distributed data storage network of thousands of machines and then running computations on that Big data using the enormous computational power of the available nodes.W.P. Jing (2015) uses Hadoop framework and clustering algorithm to analyze historical trajectory data of taxis and uses these insights to provide recommendations to existing taxi drivers about the most optimal path to take for their next journey. The paper proposes that their method of using real time GPS data of experienced drivers could provide the most optimal and cost effective route recommendation to the prospective taxi drivers. Yu et al. (2016) proposes k-nearest neighbour algorithm for predicting short term traffic condition for multi-step time intervals which includes forecasting for not just the next one time interval but more than one time interval as shown in figure 2.2 below

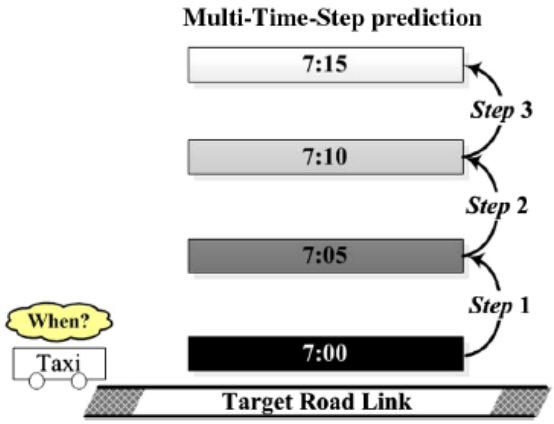

Fig. 2.2: Multi-time-step prediction (Yu et al. (2016))

Recall from the earlier section in this chapter that Euclidean distance is a distance matrix which measures straight line distance between any two points in Euclidean space using Pythagorean theorem. The k nearest neighbor algorithm searches for the most optimal nearest neighbor between the historical data and the current data and then uses

the newly found neighbors to predict future time series traffic data. K-nearest neighbor is a non-parametric test used primarily for either regression or classification tasks in machine learning. For multi-time-step prediction in figure 2.2, k-NN algorithm uses regression method to forecast future values using the nearest neighbors calculated from the historical data. The main idea behind K-NN algorithm's approach in predicting short term traffic conditions is to assign weights to the nearest neighbors calculated earlier, thus the neighbors which are nearer are assigned higher weights as compared to the neighbors further away. Predicting short term traffic congestion statistically using a machine learning algorithm like K nearest neighbors is a very powerful way to reduce urban traffic congestion and take necessary steps to improve the traffic flow especially during the current times when ride hailing services like Uber and Lyft have contributed towards increased traffic flow on the roads

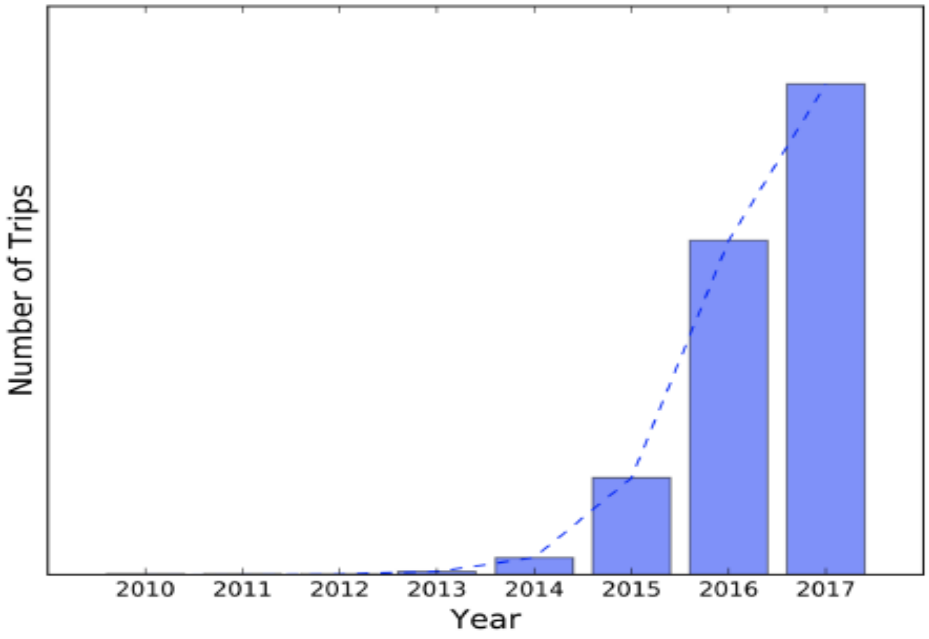

Fig. 2.3: Uber's annual trip growth (Korolko et al. (2018))

Chapter 3

Model

Finding the next profitable and economically viable passenger request is a difficult process for Uber drivers and with the recent explosive growth in demand for Uber's ride hailing services, drivers prefer software guidance to get routed to the high demand geographical locations. Guiding drivers to the next most profitable passenger is one of the most crucial factors in determining the success of ride hailing services application software. This would increase capacity utilization by reducing idle travelling time of drivers and also help in minimizing rider's waiting time for drivers to pick them. Therefore, we are proposing in this research that in order to facilitate Uber drivers search for their next profitable customer and to minimize their random cruising or empty driving, we start by dividing a city, e.g. Bay Area into a number of clusters and use machine learning algorithms like Poisson distribution and Auto Regressive Integrated Moving Average (ARIMA) statistical methods to forecast expected demand in each of these clusters

Using the expected future passenger demand for ride hailing services in the next few time periods, we would use optimal route allocation algorithm to dynamically allocate drivers by also calculating the expected profit associated with the respective demand forecast for a certain geographical region. This advanced driver optimal allocation would guide drivers beforehand about their next ideal cruising strategy to facilitate an upcoming passenger ride hailing request which would minimize driver's idle travelling

time. Therefore, the two main assumptions behind developing a model for this research include the set of N potential geographical clusters denoted by $S = \{s_1, s_2, \ldots, s_N\}$ and a set of j possible passenger destinations represented by $D = \{d_1, d_2, \ldots, d_j\}$. Also, each of our time period is of 30 minutes so there are 48 time periods in a day. Recall from the Introduction section that the main objective is to minimize the imbalance between supply and demand for Uber drivers so that driver's idle travelling time as well as passenger's waiting time are reduced

Our current optimization problem involves choosing one of the best cluster areas in the city in terms of profits for the drivers at time instant t so that once current passenger is dropped, the driver has the ability to get directed to the next available profitable passenger in the least amount of time. This is based on the forecasts made about the future possible passenger demand using the available time series forecasting methods including Poisson distribution and Auto Regressive Integrated Moving Average (ARIMA) model. This is presumed to be a valuable information for Uber drivers as it would enable them to avoid random cruising strategies to look for their next passenger pick-up and drop-off and thus provides optimum profit and reliability for all the stakeholders. Thus, the data mining on historical data plus the forecasts made for future data points enable drivers in making smart driving decisions to better facilitate passenger demand as well as optimizing the resources of Uber

Let $X = \{X_{k,0}, X_{k,1}, \ldots, X_{k,t}\}$ denote a discrete timeseries for the number of demanded services at a particular area in the city represented by a certain cluster in our model. The main goal of this mathematical model is to predict the set of service counts $X_{k,t+1}$ for instant t+1 and for each cluster $k \in \{1, \ldots, N\}$. We are proposing three main time-series forecasting methods for building our model and to attain the forecasted values of passenger demand. As passenger demand for ride hailing services

varies periodically with changing time periods and is also affected by social event line concerts, games, holidays etc, companies like Uber incorporate surge pricing to match changing customer demand for drivers. The surge pricing methodology takes precedence when the demand for Uber services is higher than the number of available drivers as shown below

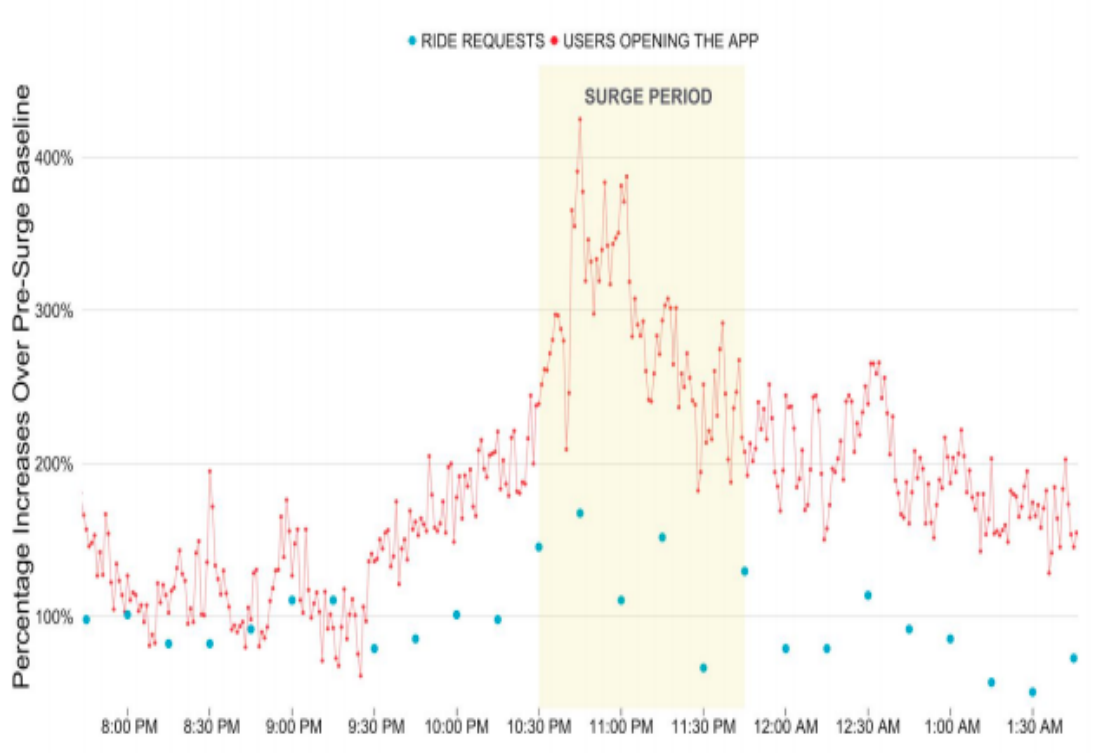

Fig. 3.1: Riders opening Uber app after a concert at Madison Square Garden (Hall et al. (2015))

The above figure 3.1 shows a sudden increase in the number of people opening Uber application following a concert and checking the price for their desired travel requirements. The red line shows the number of people opening the app while the blue dots represent the average sum of rides requested in 15 minutes interval during the same time period. Since the demand for Uber drivers increased than the usual demand , the effects of surge pricing was reinforced by Uber to charge higher prices for the sudden

surge in demand

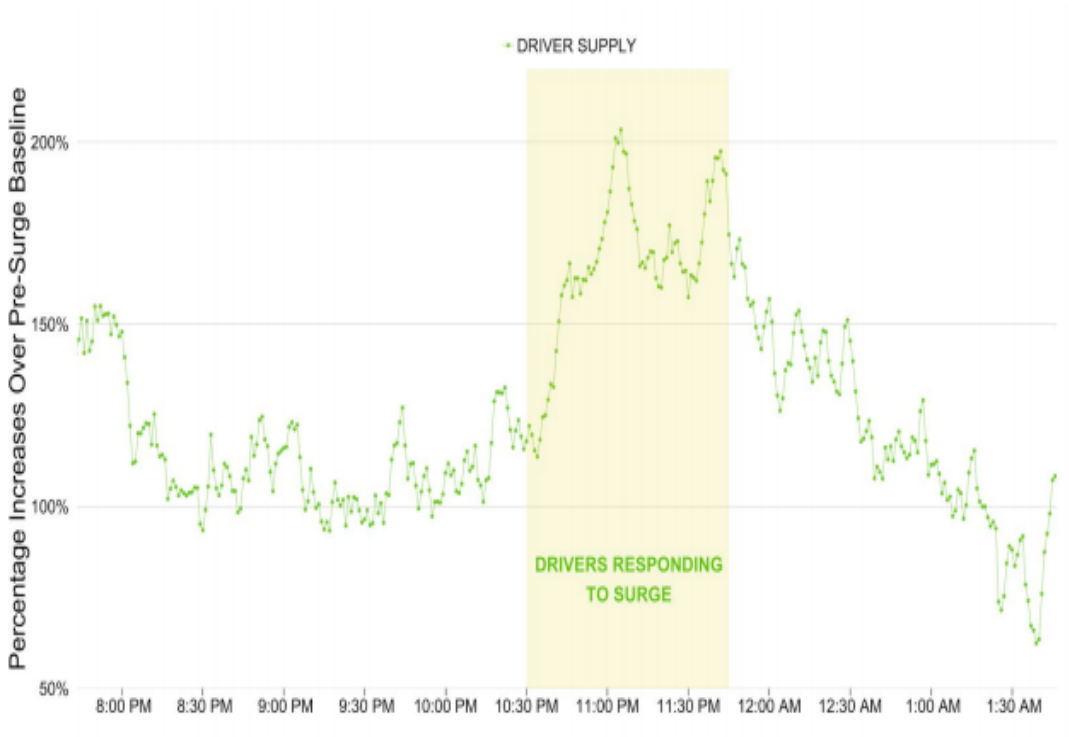

Fig. 3.2: Uber driver supply to match a spike in passenger demand after a concert at Madison Square Garden (Hall et al. (2015))

The above figure 3.2 shows the supply of Uber drivers to match sudden increase in passenger demand following a concert at Madison Square garden. Such sudden spikes in passenger demand for Uber related services is an important factor in determining the efficiency of driver routing optimization algorithms. By using dynamic pricing strategies and optimal allocation of drivers to match passenger requests, we would show that it would help in reducing the waiting time for both riders and drivers besides providing increased capacity utilization, trip throughput and the total social benefit for all the stakeholders. However, in this research we have not incorporated dynamic pricing or surge pricing strategies in our algorithm. Our research is mainly focused on finding the most optimal allocation of drivers to the most profitable geohashes to facilitate

passenger ride hailing requests in the next time periods of each 30 minutes using the forecasted passenger demand and the expected revenue in each of those locations. The three proposed time-series forecasting methods being used to calculate expected future passenger ride hailing services demand are described as follows:

3.1 Time-Varying Poisson Model

This model is an extension of previous research on modelling of freeway traffic and building security logs with Poisson distribution Ihler et al. (2006) as well as Moreira-Matias et al. (2012) which focused on analyzing passenger demand for taxi service over a period of time. Recall that Poisson distribution is a discrete probability model,which means that the occurrence of an event is always a whole number, that calculates the expected number of times an event is likely to occur within a specified time interval. Poisson Model is primarily used for predicting the probability of independent events occurring at a constant rate in a fixed interval of time or space. Based on this, we are proposing that the demand for Uber services varies periodically and is based on daily, weekday, weekend and social events like league games. Therefore, facilitating changing passenger demand on the basis of daily periodicity is a big concern for companies like Uber and Lyft. Their most important goal is to increase ethical profit margins besides ensuring highest possible standards of customer satisfaction and customer service. The assumptions of Poisson based statistical distribution are as follows:

- K is the number of times an event is likely to occur and k can take any values 0,1,2....
- All events are independent of each other such that the occurrence of one event does not affect the probability that the next or any event beyond that will likely

occur as well

- The average rate at which events occur is constant

Now using these assumptions to model the demand of Uber passenger services, we are interested in calculating the probability of such demand across different time periods using the historical data. Let's assume that the probability of n Uber services to emerge in a certain time period P(n) , follows a Poisson distribution as represented by the following equation:

$$P(n;\lambda) = \frac{e^{\lambda}\lambda^{n}}{n!} \tag{1}$$

λ represents the average rate of demand for Uber services in a fixed time interval and is therefore a function of time $\lambda(t)$. However, λ is not constant but time variant and hence changes periodically based on the time of day and is also influenced by holidays and special events thereby transforming the Poisson distribution into a non-homogeneous one. Suppose that λ_0 represents the rate of change in Poisson process for the demand of Uber services over one week. Therefore, $\lambda(t)$ is defined as follows:

$$\lambda(t) = \lambda_0 \delta_{d(t)} \eta_{d(t),h(t)} \tag{2}$$

$d(t)$ represents values from 1 to 7 and indicates the day of week for time(t) so that Sunday =1, Monday=2, Tuesday=3, Wednesday=4 and so forth. $\delta_{d(t)}$ is the day effect for a weekday d(t), e.g. Saturdays have lower day rates than Tuesdays. $\eta_{d(t),h(t)}$ is the relative change for period h(t) in day d(t), h(t) represents the period of time t such that each period is of 30 minutes and thus a day of 24 hours constitutes 48 periods

Equation (2) requires the validity of the below equations:

$$\sum_{i=1}^{7} \delta_i = 7 \tag{3}$$

$$\sum_{i=1}^{I} \eta_{d,i} = I \quad \forall d \tag{4}$$

I denotes the total number of intervals in a day. The result is a discrete time series $\lambda(t)_k$ which represents Uber services passenger demand of a particular cluster in the city during an entire week. Each value in this timeseries is calculated using average of all previous passenger demands on the same day and during the same time period during that day. For e.g. the expected Uber passenger service demand for a Tuesday from 10:00 to 10:30 is the average of the demand on all previous Tuesdays from 10:00 to 10:30

3.2 Weighted Time Varying Poisson Model

The previous model is dependent on historical data and focuses on time dependent passenger demand for Uber services to predict the expected service demand in future. However, the Weighted Time-varying Poisson process also incorporates the seasonality effect in the model as in certain cluster areas of the city, e.g. beaches,parks and stadiums, the demand for Uber services may be higher during summer weekends as compared to other seasons of the year or as shown in figure 3.1 the demand for Uber drivers increased substantially after a concert at Madison Square garden. While calculating expected passenger demand across different geohashes, we need to keep in

mind that every geohash will not have a highly regular passenger demand. Infact, the passenger demand in many geohashes will be seasonal and that's where Weighted Time varying Poisson Model could be very useful. Other use cases include special occasions like super bowl games when people flock to stadiums in masses and thus exhibit sudden spikes in demand for transportation and ride hailing services. Weighted Time varying Poisson Model takes care of such seasonality effects by assigning decreasing weights to past observations using exponential smoothing which is a time series forecasting method used for uni variate data that exhibits trends and seasonality component. The main idea behind this model is to use a weighted average method and thus assign higher weights to recent patterns of service demand (e.g., weight assigned to the Uber passenger demand for last Monday is higher than the weight assigned to the Uber passenger demand for two or three Mondays earlier). In other words, the most recent the observation, higher is the weight assigned to it. Similarly in our model, the weight ω is calculated using exponential smoothing for time series data and it assigns exponentially decreasing weights over time. ω is defined as follows:

$$\omega = \alpha * \{1, (1-\alpha), (1-\alpha)^2, \ldots, (1-\alpha)^{\gamma-1}\}, \gamma \in \mathbb{N} \tag{5}$$

γ represents the number of historical periods, α is a user defined smoothing parameter where $0 < \alpha < 1$. Therefore, the resulting weighted average $\mu(t)_k$ is defined as follows:

$$\mu(t)_k = \sum_{i=1}^{\gamma} \frac{X_{t-(\theta * i)} * \omega_i}{\Omega}, \Omega = \sum_{i=1}^{\gamma} \omega_i \tag{6}$$

θ represents the total number of time periods contained in a week

3.3 Auto Regressive Integrated Moving Average Model (ARIMA)

The previous two models are based on Poisson distribution and assume the existence of seasonality with periodic and temporal change in Uber passenger demand at each of the clusters in a metropolitan location like Bay Area. Basically, those models are heavily dependent on the notion that the demand for Uber services on a particular weekday during a certain time period is highly identical to all the previous historical demands on the same weekday and during the same time period. For e.g., the demand at one Bay Area geographical cluster on a regular Wednesday will be very similar to the demand accumulated during the same period on previous Wednesdays. However, this is not always necessary, the expected demand can have possible distinctions during the same time periods at a particular location and thus we need a model that can account for this non-stationarity in the time-series data. A non stationary time series data is one whose statistical properties such as mean, variance and auto-correlation are not constant over time. This is because of the presence of trend and seasonality components in the time series data and thus it needs to be adjusted by removing the trend, noise and seasonality components using a method known as differencing before using ARIMA method to make any passenger demand forecasts

ARIMA statistical model is used predominantly for analyzing time series data and to make forecasts using the historical data to predict future data values such as the expected passenger demand for Uber services in this research thesis. ARIMA statistical model uses number of differencing transformations in order to make non stationary

time series data into stationary data(time series without trend and seasonality) before using the data to forecast future data points. Stationary time series data is one whose statistical properties such as mean, variance and autocorrelation are all constant over time and thus it helps in generating better and more accurate predictions. This is an important step in preparing the data to be used in an ARIMA model. The number of differencing transformations required to remove non stationarity in time series data is denoted by d in ARIMA(p,d,q) notation. ARIMA models have the capability to cater to such non-stationary features of our data and thus can account for the distinct periodicities in Uber services demand for different geographical clusters in our research. Auto regressive integrated moving average (ARIMA) model is generally used for univariate time-series data which comprises of only one variable (Uber passenger demand in our research) modeled sequentially over equal time periods (30 minutes of each time period in our model) and helps in analyzing as well as forecasting future data points based on the observed historical data. The future value of a variable in ARIMA model is dependent linearly on historical observations and random errors. The AR part of ARIMA signifies the fact that the main variable of interest is regressed on its own past observations. The MA part shows that the regression error is a linear combination of error terms that occurred in the past and the I part in ARIMA indicates that the data values are replaced with the difference between their values and the previous values. The process that generates the time series for Uber passenger service for a particular geographical cluster k is given as follows:

$$R_{k,t} = \kappa_0 + \phi_1 X_{k,t-1} + \phi_2 X_{k,t-2} + \ldots + \phi_p X_{k,t-p} + \varepsilon_{k,t} - k_1 \varepsilon_{k,t-1} - k_2 \varepsilon_{k,t-2} - \ldots - kq\varepsilon_{k,t-q} \quad (7)$$

$R_{k,t}$ and $\{\varepsilon_{k,t}, \varepsilon_{k,t-1}, \varepsilon_{k,t-2}, \ldots\}$ represent the actual values at time period t and the Gaussian white noise error terms observed in the past, respectively. $\phi_l(\, l = 1, 2, \ldots, p)$ and $\kappa_m(\, m = 0, 1, 2, \ldots, q)$ are the model parameters/weights, p and q are positive integers and referred t as the order of the model. For a non seasonal ARIMA model, it is defined as ARIMA(p,d,q) where p is the order (number of time lags) of the auto-regressive model, d is the degree of differencing (number of times past values are subtracted from the data) and q is the order of the moving-average model

However, the type of univariate time series data that we have in our research problem has both the seasonal and trend components, therefore to deal with seasonal data we need to use the seasonal ARIMA model also known as SARIMA. Therefore, in order to incorporate this seasonality in ARIMA model, it is represented as ARIMA(p,d,q)$(P, D, Q)_m$ where m indicates the number of periods in each season. P,D,Q show the values for auto-regressive, differencing and moving average terms for the seasonal part of the ARIMA model. Seasonal Autoregressive Integrated Moving Average (SARIMA) requires careful configuration as we need to use grid search to select the most optimum hyperparameters for both the trend and seasonal components. Higher the accuracy of those selected parameters, better will be the forecasted values of expected passenger demand. The hyperparameters of SARIMA trend and seasonal components are defined as follows Jason (2018):

The three trend elements that are same as in ARIMA are:

p : Trend autoregressive order

d : Trend difference order

q : Trend moving average order

The three seasonal elements that are not part of ARIMA but are introduced in SARIMA to deal with the seasonal elements are:

P : Seasonal autoregressive order

D : Seasonal difference order

Q : Seasonal moving average order

m : Number of time steps in a single seasonal order

3.4 Sliding Window Ensemble Framework

The previous three methods focused on learning from long-, mid-, and short-term historical data in order to predict the future expected demand for Uber ride hailing services. However, Ensemble framework is expected to be more versatile and robust than the previous methods as it incorporates the predictive performance capabilities of multiple time-series forecasting models. Ensemble method combines the effects of the last three methods namely Poisson distribution, SARIMA and Weighted Poisson time varying models in order to improve the prediction accuracy of the forecasted passenger demand for Uber ride hailing services. Suppose $M = \{M_1, M_2, \ldots, M_z\}$ represents a set of z models of interest used to formulate a time series and suppose $F = \{F_{1t}, F_{2t}, \ldots, F_{zt}\}$ represents a set of forecasted values for the next period on time interval t by those models. Therefore, the ensemble forecast E_t is defined as follows:

$$E_t = \sum_{i=1}^{z} \frac{F_{it} * (1 - \rho_{iH})}{\Upsilon}, \Upsilon = \sum_{i=1}^{z} (1 - \rho_{iH}) \tag{8}$$

ρ_{iH} is the error of model M_i in the periods contained on the time window $[t-H, t]$ where H is a user defined parameter for the size of the window. With the continuous flow of data points for the next periods t,t+1,t+2,. . . , the user defined window will also shift forward in order to determine the performance of the model in the last H time periods. This model performance is evaluated using a statistical measure called symmetric mean percentage error(sMAPE). An H sized sliding window is used to measure the model error before each new prediction is determined for the expected future passenger demand for ride hailing services. Symmetric mean absolute percentage error (sMAPE) is an accuracy measure based on the percentage errors or relative errors. The sMAPE is defined as follows:

$$sMAPE = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2} \tag{9}$$

- A_t represents actual value
- F_t represents forecasted value

The absolute difference between A_t and F_t is divided by half the sum of absolute values of the actual value A_t and forecast value F_t. The value of this equation is added to each of the fitted points t and then divided again by the number of fitted points n. sMAPE is one of the most widely used statistical measure to calculate accuracy of forecast values and offers great advantages including scale independence and better handling of outlier values. One downside of sMAPE is that if actual value or forecast value is 0, then sMAPE error values will become too large and can possibly reach an infinite value as well.

Chapter 4

Data Acquisition and Preprocessing

The stream event real-time data of UberX and UberBlack drivers is collected in San Francisco, California and records multiple variables of an Uber driver facilitating passenger's ride hailing requests across multiple locations in the city. San Francisco is the cultural, commercial and financial centre of northern California and is the 13th most populous city in the United States. Ride hailing demand for Uber services in San Francisco is highly concentrated in and around the downtown SF area during the usual business hours while North Beach, Marina and Mission are popular high demand areas for Uber passengers particularly on nights and weekends

A 2017 study found that on a typical weekday in San Francisco, ride hailing service providers (drivers) make around 170,000 vehicle trips in fulfilling passenger's demand requests. These trips are mainly concentrated in the areas where population density is higher during that time especially the downtown and financial centre areas. Also, notable attraction in San Francisco are Fisherman's Wharf, Golden Gate bridge and Golden Gate museums and parks around which we can see in figure 4.1 that the traffic congestion due to Uber's ride hailing services is significantly higher. This indicates the fact that passenger's ride hailing demand requests for UberX and UberBlack is higher in these areas. The increasing intensity of colors or the dark colors in figure 4.1 represent increasing traffic congestions in the marked areas. According to San Francisco County Transportation Authority (SFCTA), ride hailing services such as Uber and Lyft have

contributed to approximately 50 percent rise in traffic congestion in the San Francisco city between 2010 and 2016 and account for 36 percent of delay in the downtown area

The study indicators of this vehicle congestion include vehicle hours of delay, vehicles miles travelled and average speed of driving in the city. However, Uber claims to reduce traffic congestion through ride hailing services by enforcing the surge pricing methodology which basically increases the price during times of high demand. The effects of the surge pricing phenomena are yet to be seen in reducing traffic congestions especially during peak demand periods and at high population density areas like downtown San Francisco and Manhattan in downtown New York City. Uber and Lyft also claim to increase capacity utilization rates of drivers versus taxi drivers, however, in highly congested areas as mentioned above, the capacity utilization rates are competitively close to taxi drivers utilization rates.

4.1 Data Acquisition

The dataset used in this study is collected continuously over a period of time by using the iot devices like GPS sensors installed in each of the 41 unique Uber driver vehicles used in facilitating passenger's ride hailing requests. Data is collected for 15,281 different geohashes in San Francisco where each geohash is a small geographical cluster or a tiny square representing a particular area in the city. Time period of the dataset spans from May 5,2014 to March 7,2017 during which the data is collected for the Uber drivers facilitating passenger ride hailing requests by picking them from their current location and dropping them off to their desired destinations

Each data point or row in the dataset has the following nine columns or attributes: 1) ***pickup_ date***, which records the date of the passenger pickup by Uber driver; 2)

Fig. 4.1: Uber's effect on increased traffic congestion in and around downtown San Francisco (UberSF (2018))

pickup_time, which mentions the time in hours, minutes and seconds format of the passenger's pickup by the driver: 3) ***pickup_weekday***, which is the day of the week on which the event occurred; 4) ***geo_hash***, which is the geographical location of the event; 5) ***driver_uuid***, which is the unique id assigned to each of the 41 Uber drivers in this dataset; 6) ***total***, which is the total revenue from the Uber driver's trip in fulfilling passenger's ride hailing request; 7) ***distance***, which measures the total distance travelled in miles by Uber driver between passenger's pickup point and the drop-off address; 8) ***pickup_address***, which is the location from where Uber driver picks up the passenger requesting a ride hailing service; 9) ***dropoff_address***, which is the drop off location that passenger is taken by Uber driver during the ride hailing service. The dataset

has 30,555 records and is given to us in Microsoft Excel format which requires further data cleaning and data wrangling that we performed in R statistical tool. The dataset used in this thesis provides information about drivers picking passenger from different locations, the date time and weekday of the passenger pickups as mentioned below in the following table which lists all the variables and their description:

Table 4.1: Data set Key Variables, Descriptions, and Measures

Variable	Description
pickup_date	date on which driver picked up a passenger in terms of d/m/yy
pickup_time	Time of passenger's pickup by driver in terms of hours:minutes:seconds
pickup_weekday	A weekday from Sunday to Saturday on which the driver picked up the passenger
geo_hash	9 units alphanumeric address of the passenger's geographic location
driver_uuid	Unique alphanumeric id for each driver available in the pool
total	Total expected profit from the ride in which driver facilitates passenger's ride hailing pickup request
distance	distance measured in miles between passenger's pickup point and the desired drop-off address
pickup_address	Passenger's pickup address in terms of street, city and postal code
dropoff_address	Passenger's dropoff address in terms of street, city and postal code

4.2 Preprocessing and Data Analysis

We started analyzing the dataset available to us in R statistical tool and performed necessary data transformation which includes creating new variables from the available ones before carrying out our data analysis on the ride hailing services provided by 41 Uber drivers in the dataset. After importing the powerful ***tidyverse*** package in R, we changed the datatypes of variables to an appropriate format such as converting ***pickup_date*** to date form from string datatype. We also converted the string variables to categorical datatype such as day, month and week of year variables by converting them to factor format as it is required for plotting the graphs

Next, we created a new variable called ***Timeperiod***, which is a timeperiod of 30 minutes each thus representing each day by its respective 48 timeperiods to denote ride hailing activity between Uber driver and passenger requesting the ride service. To create the variable timeperiod, we first separated the existing variable ***pickup_time*** into hours, minutes and seconds components using the ***lubridate*** package in R and then used the hms function. We then developed a mathematical logic to divide the time into subsequent timeperiods by using the following statistical operations: i) First, we multiply the hours component by 2; ii) then subtract the minutes component with 30 and assign 2 if the difference is greater than 0, otherwise assign 0; iii) Third step is to again use the minutes component and subtract it with 30 and assign 1 if the difference is less than 0, otherwise assign 0. Subsequently we take the absolute sum of the three steps mentioned above to calculate the new variable timeperiod

We also assigned numeric ids to ***driver_uuid*** which was in alphanumeric format and created a new variable called ***Driver_ID*** which is in numeric format. We also performed string operation on the variable ***geo_hash*** to only use the first 5 characters

to represent each geohash for simplicity of reference. Subsequently, we used the ***dplyr*** package to use the group by and summarize functions in R to calculate total number of requests in each time period for each respective geohash for further analysis. For our exploratory data analysis where want to analyze each variable's relationship with the mean daily passenger ride hailing requests for each driver, we created new variables like day, month and week of year using the ***strftime*** function in R on the existing variable ***pickup_ date*** to split the date variable into its respective day, week and month components

Subsequently, we used the ***ggplot*** package in R to plot a graph for each of the newly created time component variables and their relationship with the mean daily ride hailing requests. We calculated mean daily ride requests by dividing the total number of requests with the total number of drivers available for that time during the 48 distinct time periods in a certain geohash. This is done for each of the variables day, week and month components using groupby function in R. These data visualizations are presented with their relevant explanation in the next section of exploratory data analysis. We also calculated average revenue for the Uber drivers serving each geohash throughout all the 48 distinct time periods. We first calculated the frequencies or the total number of trips completed by each of the 41 Uber drivers in our dataset, then we used the ***aggregate*** function in R statistical tool to calculate the total revenue realized by each of the drivers

Subsequently we divided the total expected revenue realized by each driver in all the geohashes by the respective number of trips conducted by that Uber driver in order to calculate the average revenue realized in the San Francisco city. According to the average expected revenue calculation defined above, we noted that the maximum average revenue of 92.25 occurred for the driver id 22. Despite the fact that driver id 18 and 27 completed higher number of trips than the driver id 22 as shown in figure 4.4,

the average revenue of driver id 22 is much higher than the average revenue of driver id 18 and 27. Apparently, as noted from the dataset, Uber drivers earned a relatively higher expected revenue for their trips to facilitate passenger ride hailing request when the ride is between Oakland and San Francisco

The barchart in figure 4.2 shows the average revenue earned by the 41 divers whom we recorded for their daily ride hailing services. The graph shows that the driver id 22 earned the highest average revenue of 92.25 while driver 3 has the second highest average revenue of 52 dollars. Also, the driver ids 18, 27 and 30 have the lowest average revenue even though the driver 27 completed almost 3000 trips in the data collected. The figure 4.3 shows the boxplot depicting the variation in the total revenues earned throughout all the trips conducted by the drivers during the timeframe of the data. We can note from figure 4.3 that the median average revenue is highest for the driver id 3 in the group who fulfilled 936 trips in the data that is collected. However, according to the barchart plotted in figure 4.4 , we can see that driver 4 and driver 27 completed the most number of ride hailing services in the collected data. Therefore, we can deduce an important finding that irrespective of the number of trips made by the drivers, higher expected revenue is not related to the frequency of trips and is thus correlated with the pickup and destination address of the ride. Also, we noted that ride hailing trip requests between Oakland and SF lead to higher average revenue for the drivers as compared to other location in San Francisco Bay Area. Therefore, some drivers who have been less frequent or who completed less number of trips than the other drivers earned relatively higher average revenue

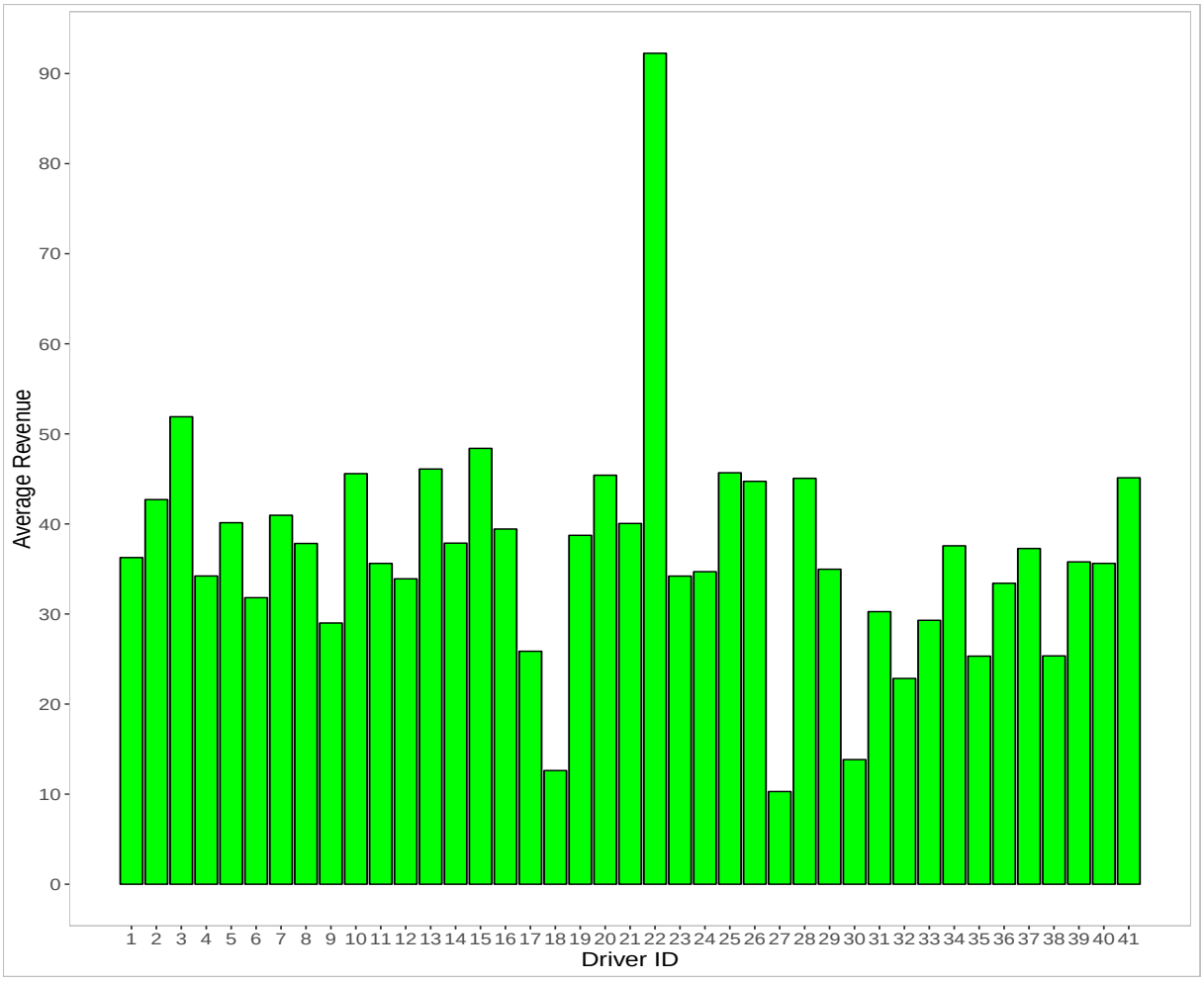

Fig. 4.2: Average Revenue for Drivers

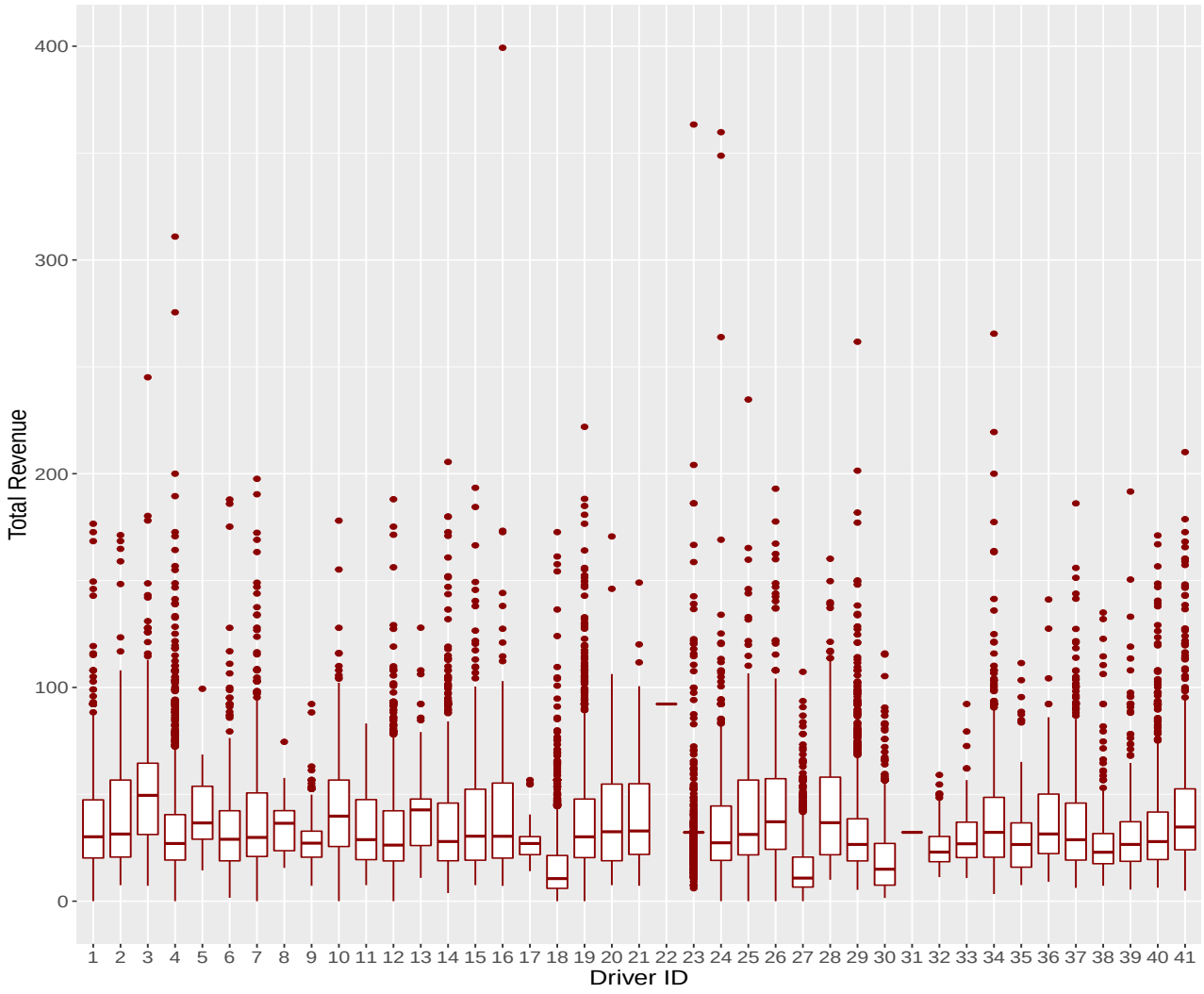

Fig. 4.3: Total Revenue for Drivers

The boxplot in figure 4.3 shows the variation in the total revenue in all the ride hailing trips carried out by all the drivers. We can see that the maximum average revenue of 399.27 is an outlier in the plot and occurred in the geohash 9q9ng1hk1 for the driver id 16. This driver facilitated passenger's ride hailing request by picking the passenger from Colesium Way in Oakland and dropping at Mission street in San Francisco. Most of the drivers have lot of outliers above the upper whisker or outside of the 75th percentile of the boxplots, this indicates that these drivers earned relatively

higher total revenues which are beyond their usual revenue earned.

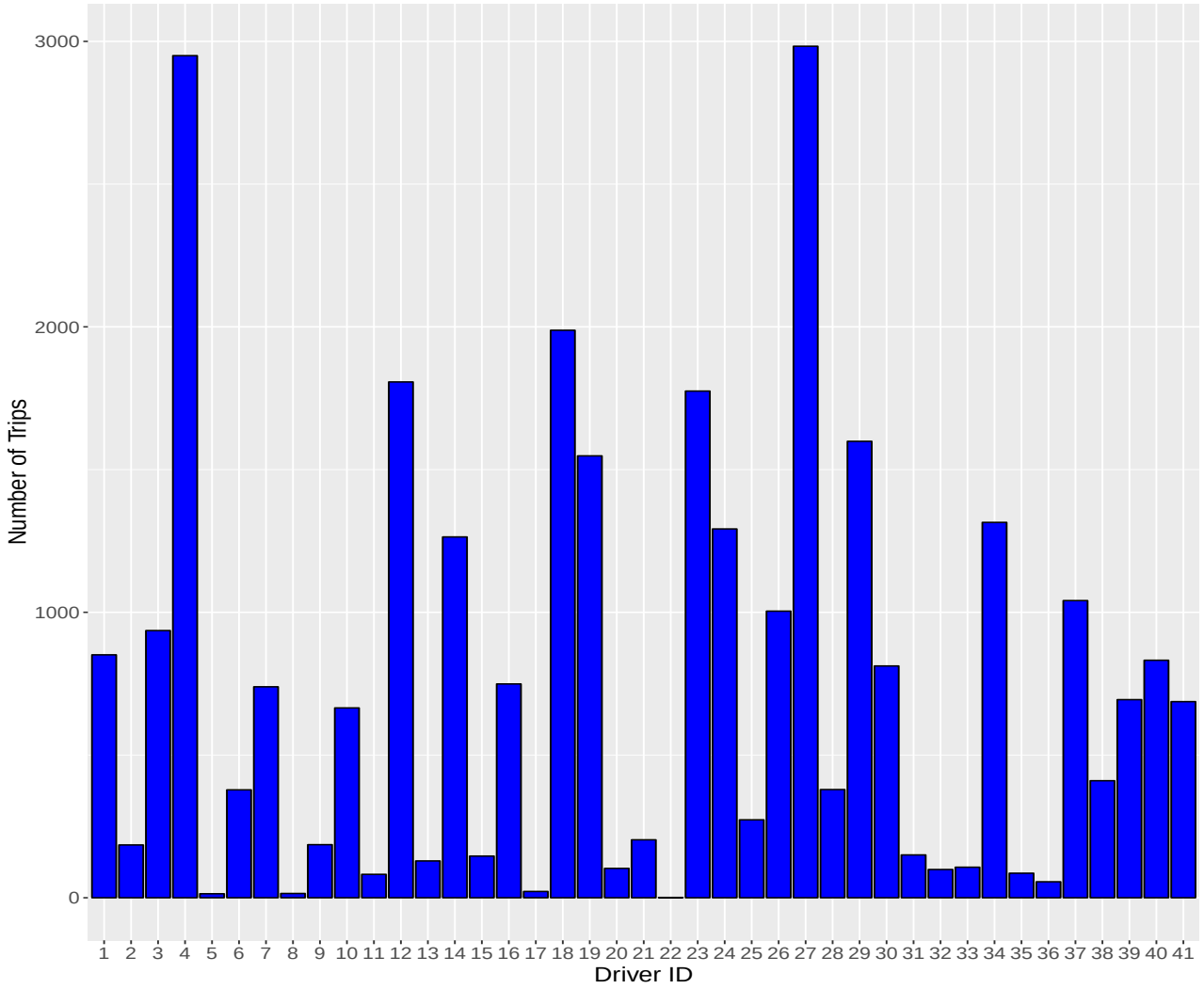

Fig. 4.4: Frequency of Ride hailing trips by the Drivers

As mentioned earlier, the bar chart in figure 4.4 just provides an overview of the total number of trips completed by the 41 drivers from the period of May 5,2014 to Mar 7,2017. Some drivers like driver id 4 and 27 have been the busiest drivers as they both completed close to 3000 trips during the period of data collection. However, the median of total revenue as well as the average revenue of driver 27 is much less than that of the driver 4.

4.3 Forecasting Methods

Recall from the Introduction section in chapter 1 that our main objective in this thesis research project is to reduce the current imbalance between the supply of Uber drivers and passenger's ride hailing demand. We want to reduce this imbalance by optimally allocating the Uber drivers in the most efficient way so that the driver's idle cruising time as well as the passenger's waiting time for the Uber vehicle to arrive at their pick-up point are minimized to the best possible extent. In order to achieve this objective of reducing demand supply imbalance , we used Machine Learning algorithms like ***Poisson Distribution*** , ***Auto Regressive Integrated Moving Average(ARIMA)*** and ***Sliding Window Ensemble Framework*** methods to calculate next period demand forecasts for t=30 minutes timeperiod intervals by first dividing our dataset described earlier in this chapter into training and test datasets so that our forecasts are as accurate as possible

Our dataset has 148 weeks of data which we then divided into 98 weeks for the training data and the remaining 50 weeks for the test set. Therefore, based on our timeperiod variable definition that each day has 48 distinct timeperiods, the training data has now 32,928 timeperiods and the test data has 16,800 distinct timeperiods of historical data to train and test our forecasting algorithms. This helps in increasing the accuracy of our predictions for expected future passenger ride hailing demand. The ARIMA model (p,d and q values, and seasonality) is found initially using grid search methodology to search the most optimal hyper-parameters by using the 2 weeks (14*48=672 time periods) of historical training data

The predictions using ARIMA forecasting method are done using the robust time-series function in the [forecast] ***auto-arima*** package in R. The weights/parameters for

each model and for each timeperiod are calculated using the ***arima*** package in R. The time-varying Poisson distribution methods including both weighted and non-weighted are also calculated using training data with 98 weeks and test data with 50 weeks. A sliding window of 4 hours (H=8) is utilized for the Ensemble forecasting method. Subsequently, a well known error measurement ***Symetric mean absolute Percentage error(sMAPE)***, defined in section 3.4 is used to evaluate the accuracy of the forecasted demand values. The results are presented as follows:

Table 4.2: Error Measured On the Forecasting Models using sMAPE

Model	sMAPE Error
Poisson Mean	70.08
Weighted Poisson Mean	80.51
ARIMA	73.29
Ensemble Framework	70.40

The errors calculated as shown in the above table 4.2 indicates the overall performance of the time-series forecasting methods being utilized in this thesis. The maximum value of error is 80.51 for the Weighted Poisson method while the top performers are Ensemble Framework and Poisson Mean methods with errors 70.40 and 70.08 respectively. Since Ensemble method incorporates the predictive performance capabilities of multiple forecasting methods, it is therefore more robust and is thus one of the best

methods as per the errors calculated using the ***sMAPE*** accuracy measure.

4.4 Exploratory Data Analysis

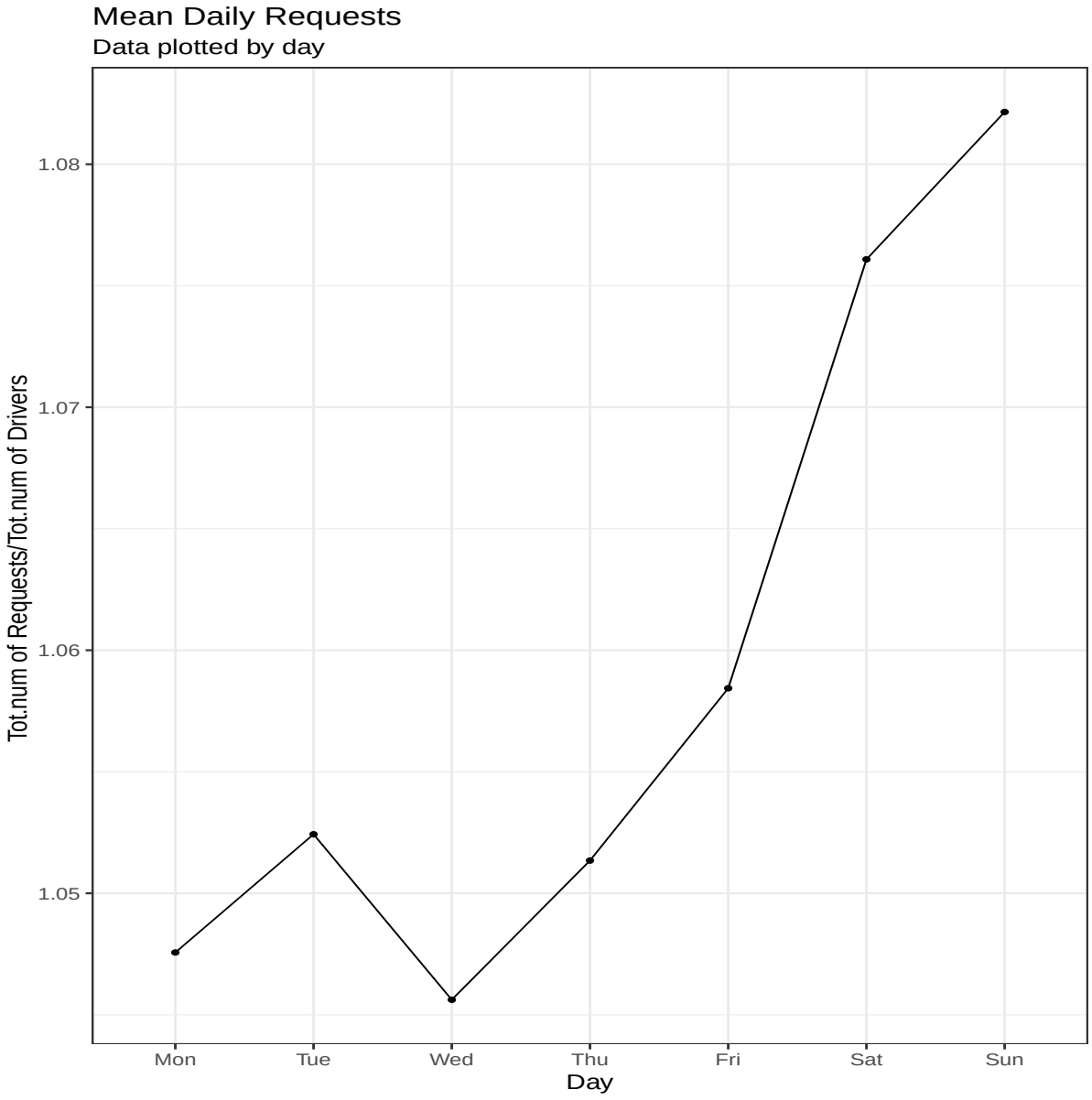

Fig. 4.5: Mean Daily Requests

The above figure 4.5 shows the mean daily requests for each day in a week where we calculated mean requests by diving the total number of requests with the available total number of drivers within that particular time window. The graph reflects the change in average requests for Uber services over different days and across all the geohashes in consideration. We can see that the demand or mean requests increases significantly over the weekend (Saturday and Sunday) while in weekdays it is much lower than the maximum threshold of 1.08. This satisfies our original assumption that people use ride hailing services more frequently over the weekend and for longer routes as during the business days they just ride between home and work. Therefore, the mean daily requests on weekdays is less that that on weekends

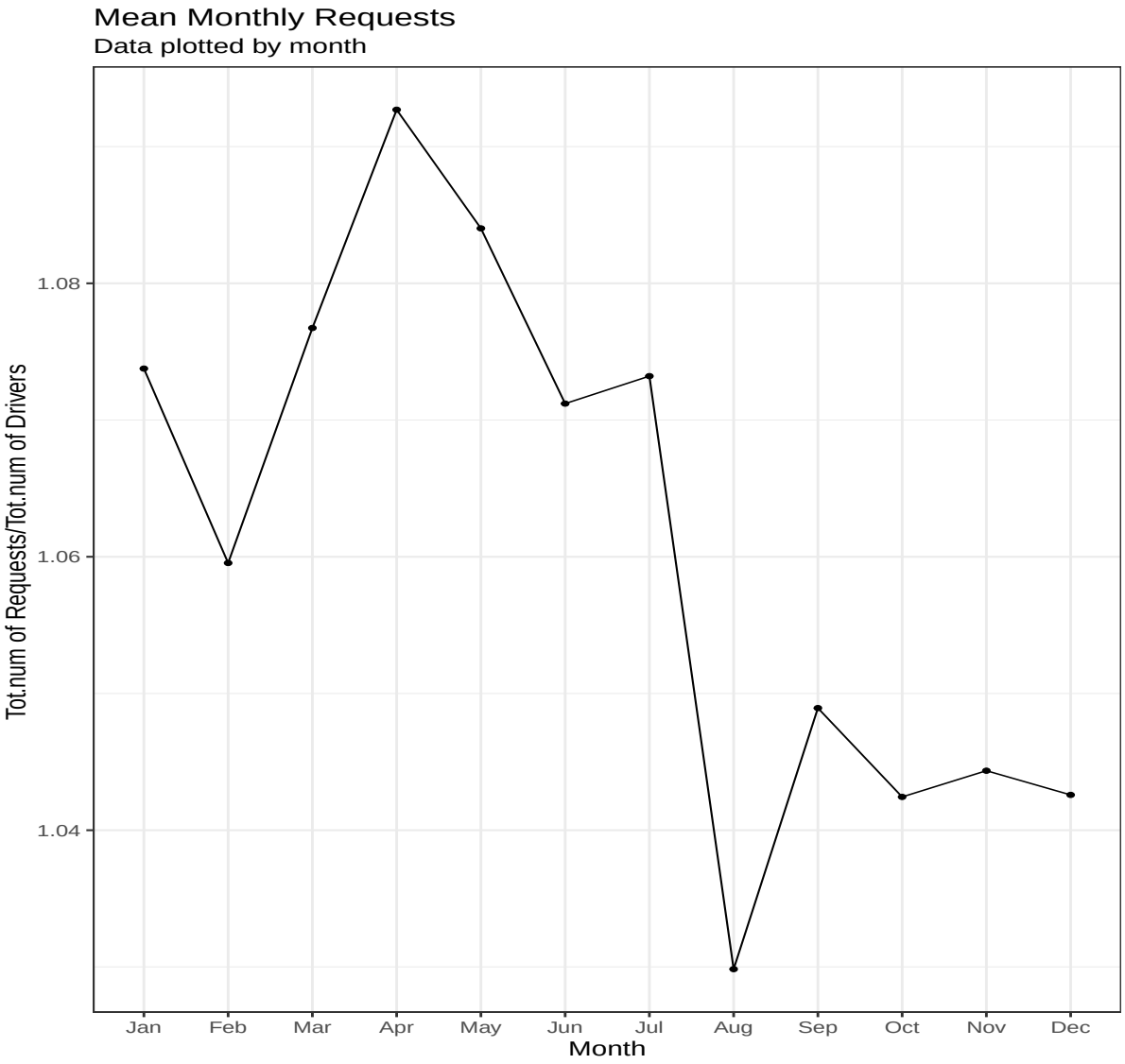

Fig. 4.6: Mean Monthly Requests

The graph in figure 4.6 shows the variatio in mean monthly requests and we can see that the mean requests calculated by dividing total number of requests with total number of drivers was highest in April and lowest in August. Based on this graph and using the available data, we can deduce a finding that the mean requests or people's propensity to request Uber services has been higher in winter and spring months and then continued to fall during summer. This can be attributed to the fact that in

summer students are on vacations and most of them travel to their homes overseas which significantly reduce the demand for ride sharing services. In San Francisco, there is a large number of international students who travel during summers and thus impact the ride hailing service industry in a sizeable way. Also, since kids are on summer vacations, so parents take them to other places which thereby affect the passenger demand for rides.

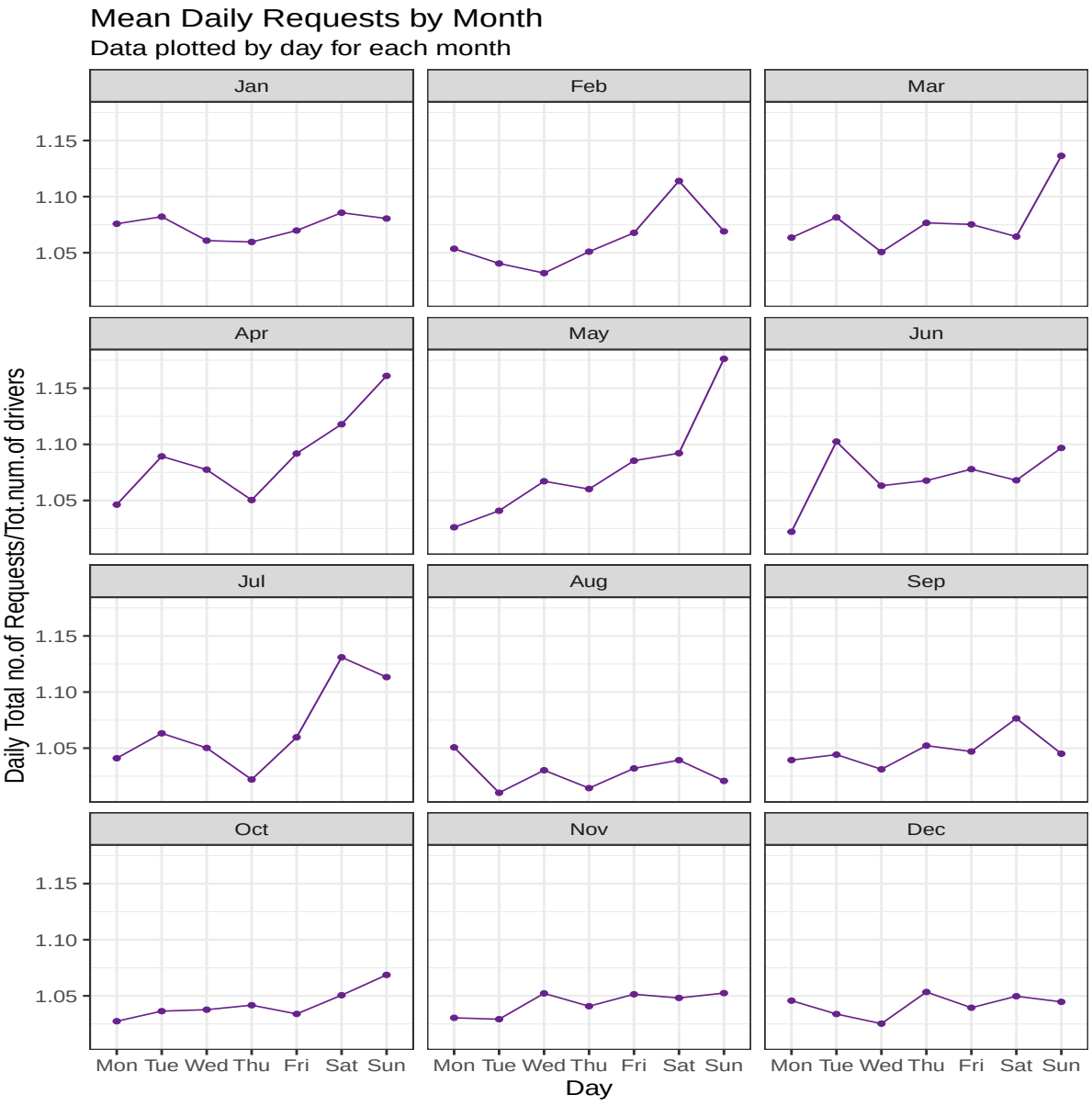

Fig. 4.7: Mean Monthly Requests by weekday

The above graph in figure 4.7 shows the combined plot for day and month and provides variations of each day within a month. This is plotted using facet wrap function in R and provides a convenient display of 2 dimensional plots. Some interesting findings from this plot include the sharp increase in mean passenger requests during the weekends in most of the months like May, June, July, August and October . This is aligned with our initial notion that demand for ride sharing services is higher on weekend days.

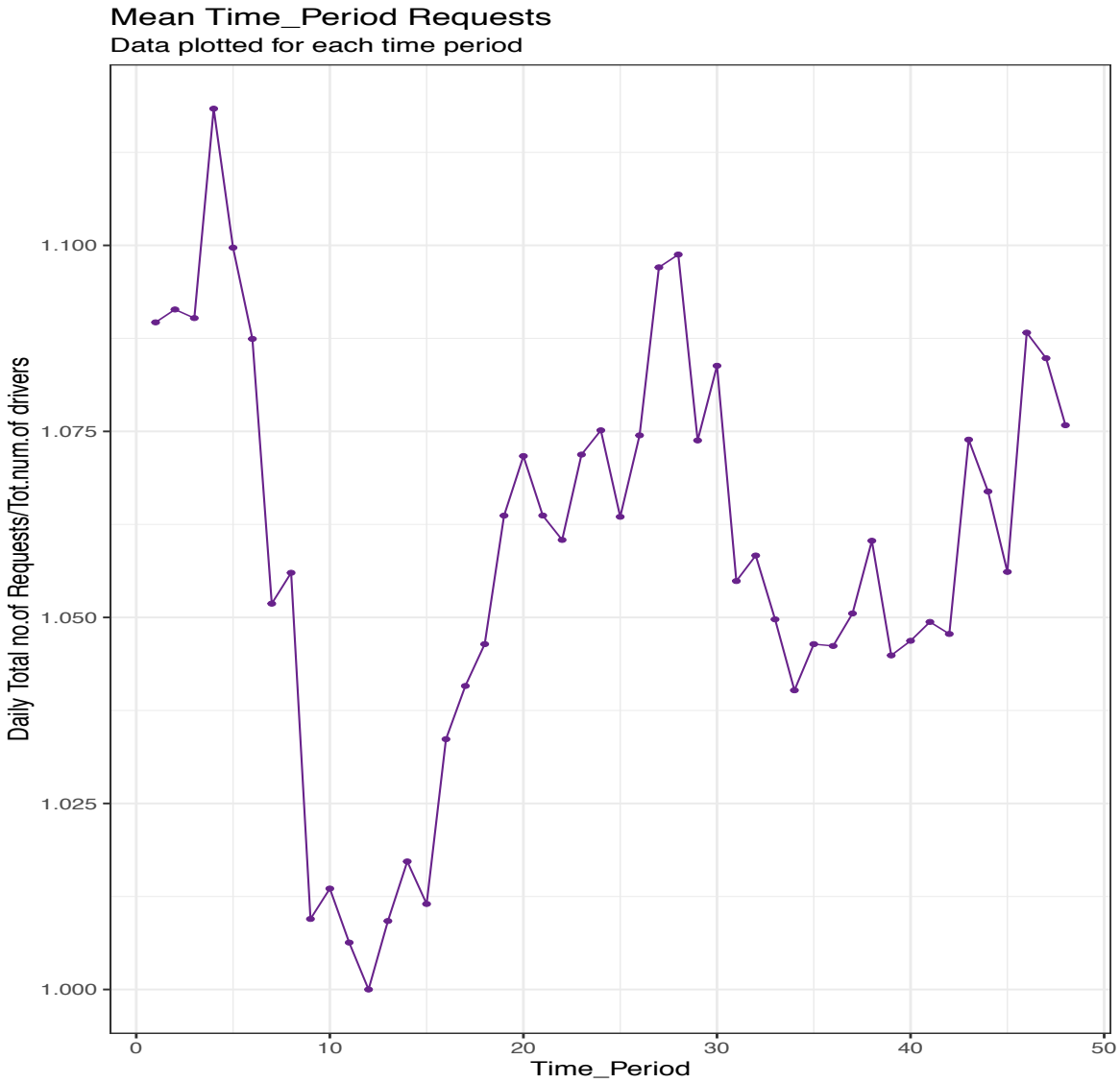

Fig. 4.8: Mean Time Period Requests

Recall that one of our defined assumptions in the optimization model is to have a time period of 30 minutes each and thus have 48 time periods in a day. The above plot is figure 4.8 shows the mean request variation with different time periods. The mean time period requests increase sharply form time period 12 on wards indicating the increase in passenger movement pattern as the day progresses from early hours in the morning and then stays higher throughout the rest of the day until midnight. This busy activity signifies the increase in demand by passengers during business hours such as going to work and coming back from work

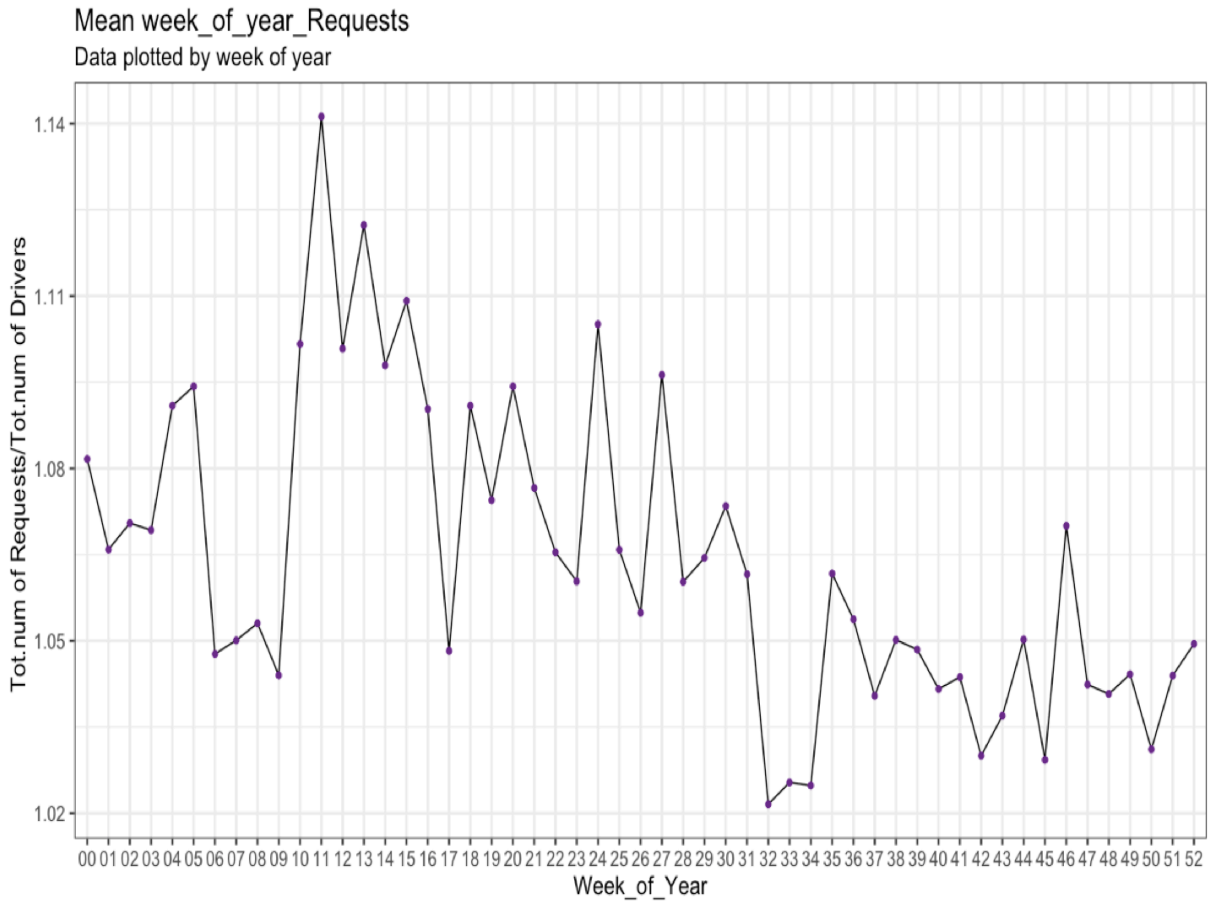

Fig. 4.9: Mean Requests per week of year

The above figure 4.9 shows the mean passenger requests for Uber ride hailing services denoting the week numbers from 0 to 52 for each of the years represented in the dataset from 2014 upto March 2017. The graph shows the average requests for each week during the mentioned time period. We can note some quick insights from the above graph including the upsurge in passenger demand during the last few weeks from week 45 to week 49 at the end of each year due to the events like Thanksgiving, Christmas and New Year's eve. Another crucial insights is the somewhat decrease in passenger demand during the summer months when students are on vaccations and mostly holidaying or travelling with family thus resulting in slight decrease in demand for ride hailing and ride sharing services

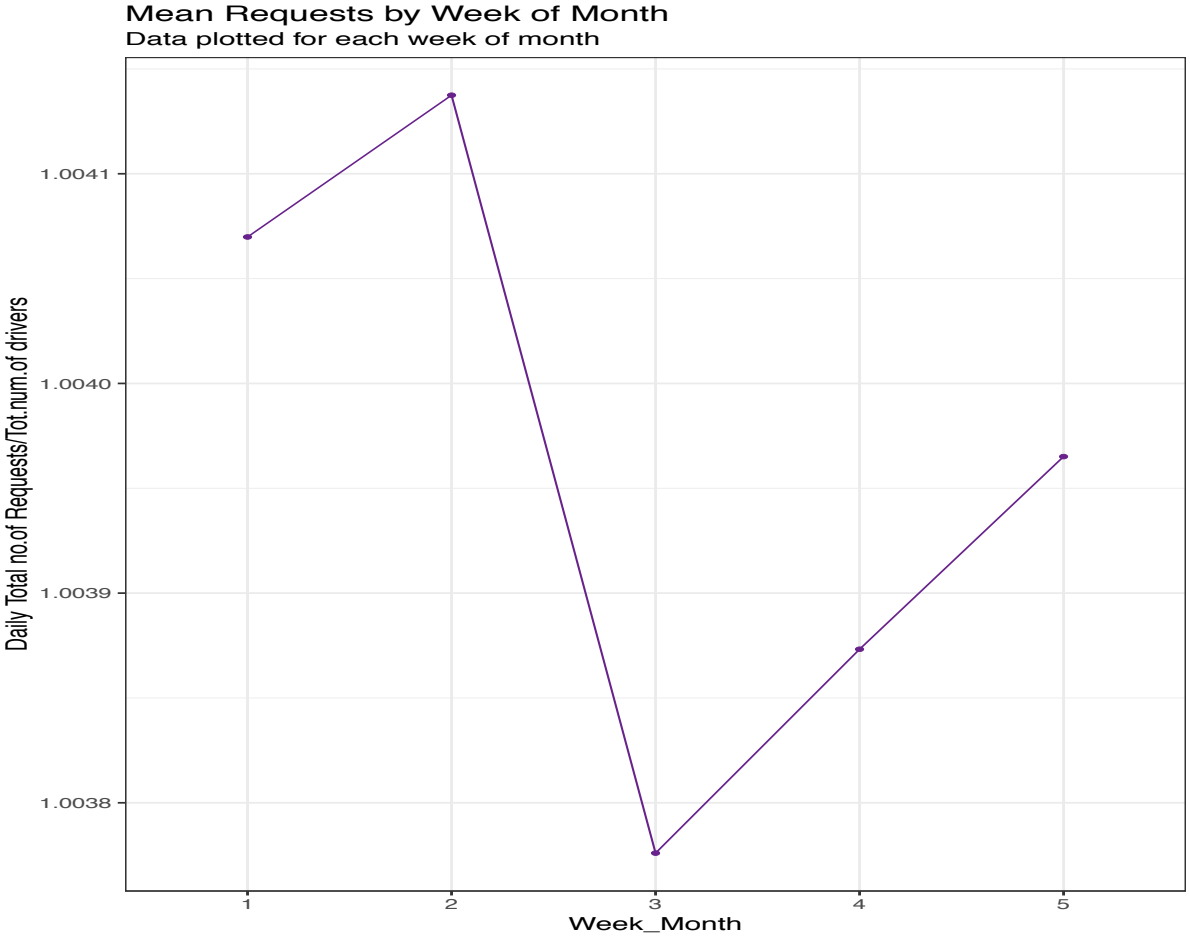

Fig. 4.10: Mean requests per week of month

The above figure 4.10 shows the mean requests for each week in a month for all the months in the dataset. The graph has an important observation showing the rise in mean requests from week 1 to week to 2 , a sharp decline in the mean requests from week 2 to week 3 and again a sharp rise from week 3 to week 4 towards the end of the month. An interesting reason behind this trend could be attributed to the disbursement of paychecks at the end of previous month resulting in higher usage by people beginng the first week of the month and again when the next paycheck comes in after a fortnight, average requests for ride hailing services increase sharply from week 3 onwards towards the end of the month. Also, the drivers are very active in meeting targets and achieving higher capacity utilization rates in the start and towards the end of the month

Fig. 4.11: Mean Monthly Requests by day of month

The above plot in figure 4.11 shows the mean daily requests for each driver among the pool of available drivers in service denoting the variation in mean requests for each numeric day of a month. There are some interesting observations in the above plot,

such as a sharp rise in mean requests after the first week of January signalling the fact that passengers including students and office workers getting back to school and work after long span of Christmas and New Year's eve holidays.This gives rise to a sudden spike in Uber ride hailing services demand reaching a high of 0.16 mean requests per driver in the first week of January as per the data available for analysis in this research

The highest observed mean daily passenger ride hailing demand request of 0.16 occurred in the first week of January as that is one of the times in a year when business activity picks up and increases sharply after a long span of holidays in the last two weeks of December. Also, the last week of December has very low mean daily requests as that's when the things slow down especially the ride hailing activity due to people having vaccations and Christmas holidays. Another finding is that the summer months like May, June and July have the lowest mean daily driver requests as people travel around due to school and college summer vaccations. Also, we can see that there is a decreasing trend in mean daily passenger requests from May to August due to low activity. The month of September has an upward increasing pattern in the daily mean driver requests as this is when students head back to classes after long summer holidays and thus inducing a rise in the usage of ride hailing services which ultimately increases the mean daily passenger ride requests for each available driver

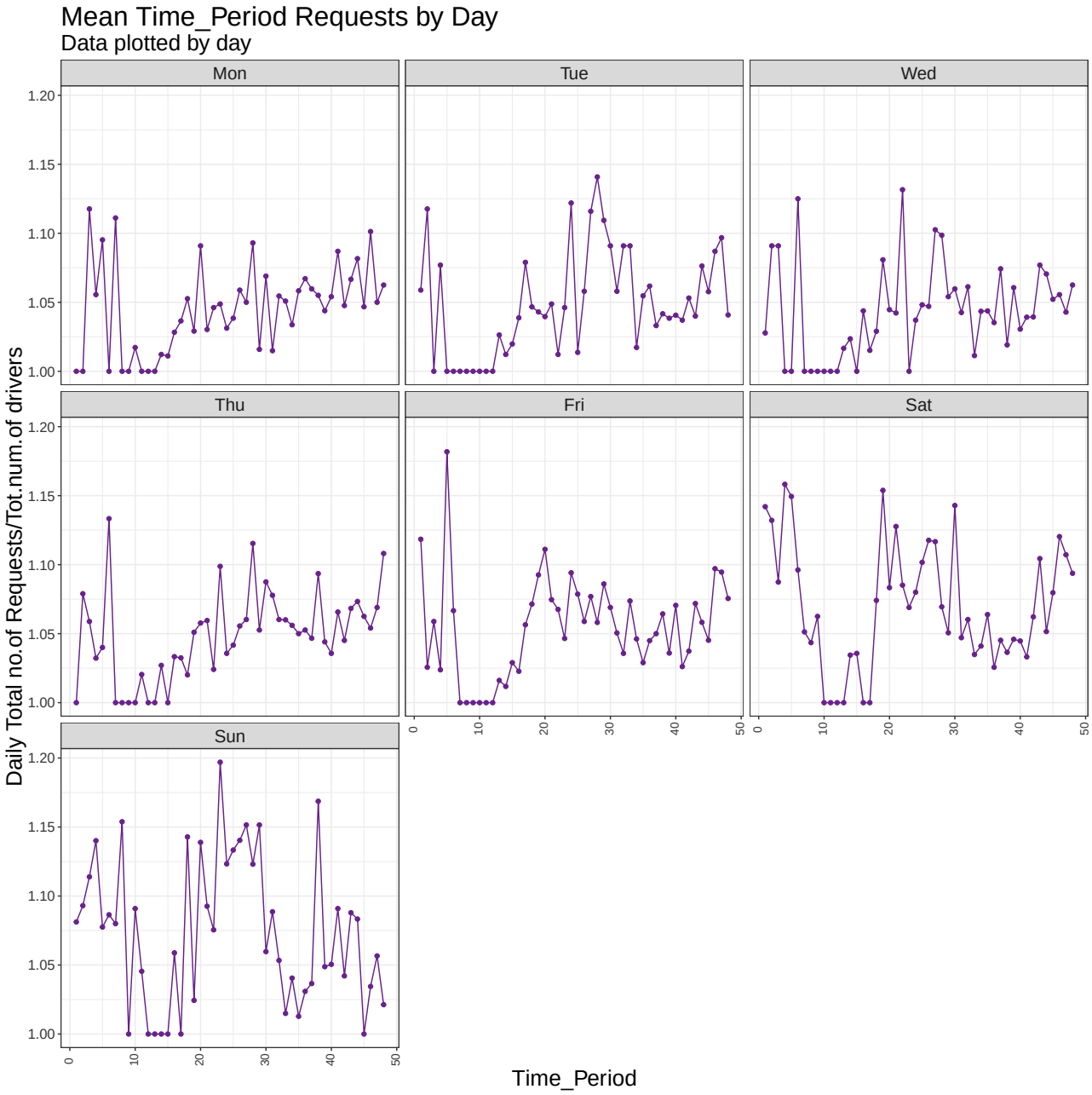

Fig. 4.12: Mean requests per Time Period of day

The above plot in figure 4.12 shows the mean passenger ride hailing requests per driver for each of the 48 time periods of a day in a week. Some interesting findings could be noted from the plot such as the lowest mean daily passenger requests of 1.0 occurred

during the time periods 5 to 15 as these are during the pre-dawn hours of the day. This is in line with our findings and assumptions that the passenger ride hailing requests are expected to be very minimal during the late night hours. During our data cleaning and data wrangling parts of our original dataset, we defined each timeperiod to be equal to 30 minutes and started a day after midnight everyday. Therefore, it is very clear from the graph that early time periods have very minimal ride hailing demand for Uber drivers. Subsequently, ride hailing demand surges drastically after time period 16 which 8 am in the morning and that's when people (office goers and students predominantly) exhibit higher ride hailing demand due to their travel requirements to offices, schools and colleges

All weekdays except Sunday exhibit very high increase in mean daily passenger requests as Sunday being holiday the early morning ride hailing demand is not as high and drastic as in other days. Another finding is that after time period 42 which is 9 pm , the mean daily passenger's ride hailing demand for Uber drivers start to die down and decrease rapidly and this decrease is highest on Sunday as that's when people have to prepare for next day start of the working week and usually sleep early on Sundays. This results in a sharp decline in passenger ride hailing demand requests during the late night hours of Sunday as compared to other weekdays during the same time periods. All of these findings are in resonance with our research assumptions and constraints defined earlier

Chapter 5

Driver Allocation Problem

We want to be able to address the problem of ***dynamic vehicle routing*** where Uber drivers are allocated and dispatched to serve a passenger demand emerging in real time during a time period of 30 minutes, thus spreading over 48 time periods in a single day. Passengers want to be able to book a last minute ride service and prefer the shortest arrival time of Uber drivers even during the busy hours of a day. Ride hailing service providers like Uber facilitate passenger requests during busy times like concerts or New Year's eve by enforcing ***dynamic pricing***. This serves the interests of drivers who get higher expected profit from serving passenger requests during such busy times as well as the interests of passengers who get efficient ride hailing services by having the shortest possible waiting time for their ride

Recall from chapter 1, that we want to reduce the waiting time of both the drivers and passengers by using optimal route optimization for Uber drivers so that their next passenger is allocated in the shortest possible time after their existing passenger is dropped off. This helps in minimizing random driving time and saving on time and fuel resources besides also increasing the ***capacity utilization rate*** of the drivers which is the amount of time a passenger occupied the Uber driver's vehicle . We also want to be able to incorporate the fact that the driver closest to the passenger demand is a better allocation than the driver further away from the passenger. Uber uses the strategy of ***dynamic route upgrade*** which automatically allocates the nearest available

driver to the passenger ride hailing request even after the initial allocation was done to another driver who is further away and is on the way. The newly allocated driver using the dynamic trip upgrade just got available and was in much closer proximity to the passenger than the first allocated driver. So, the software keeps checking the closest available driver until passenger request is facilitated by picking up. Therefore, we are addressing the existing imbalance between supply and demand of Uber drivers and aspire to minimize this imbalance through this research by developing an algorithm to optimally allocate drivers to their most profitable and economically feasible passenger locations

After dividing a city like Bay Area into several clusters(geohashes)$\boldsymbol{J}$, we propose to allocate Uber drivers to those geohash locations based on the predicted demand in terms of number of requests $\boldsymbol{q_{jt}}$ and expected revenue $\boldsymbol{R_{jt}}$. We assign every passenger with the origin and destination nodes besides assigning the drivers with their existing locations and the associated expected revenues. Through this, we aspire to ensure that driver's idle travelling time is minimized while passenger's waiting time is also at the minimum. The algorithm is designed to include the condition that it would route the drivers to the most profitable locations so that when they receive passenger ride requests, they are as close to them as possible. Therefore, we are aiming for a balanced driver allocation strategy that well serves the purpose and demand of all the stakeholders including driver and passenger as well as the driver's company Uber

Our research is also aimed at reducing the difference between actual revenue and expected revenue as this would increase the accuracy of our algorithm and its associated forecasts. This would also assist in minimizing the difference in income among Uber drivers who work equal number of hours in an adequately sized geohash location where passenger demand is decent enough for drivers to serve.We also want to take into account

the fact that a driver who is about to finish his/her ride near the demand(rider) is better than allocating the driver who is available but is far away from the passenger requesting the ride. We also want to incorporate the fact that the supply of Uber drivers is allocated in accordance with the travel intensity of each geohash cluster. Therefore, geohash locations which are busy and thus have higher expected number of passenger requests in a certain time period t, will be allocated less or equal number of Uber drivers to facilitate customer requests for ride sharing services. This is expected to keep the supply and demand of ride hailing service providers (drivers) in balance as our main objective is to address this imbalance between supply and demand of Uber drivers by using smart route optimization

The optimization problem is therefore, at time t, how should we allocate drivers across different geohashes and time periods in order to maximize driver's revenue and to reduce rider's waiting time both before getting picked up as well as in reaching their final Uber destination to the best possible extent. This introduces us to a concept of ***capacity utilization rate*** which measures the amount of time a driver has paying passenger group in car and facilitating the respective passenger's ride hailing or ride sharing request. It also measures the number of miles that the driver drove each day with a fare paying passenger in the car

Our constrained optimization driver allocation problem and its variables are defined as follows:

Table 5.1: Key Variables, Descriptions, and Measures

Variable	Description
i *(subscript)*	Driver
j *(subscript)*	Geohash location
t *(subscript)*	Time period (30 minutes each)
q_{jt}	Number of requests from geohash j at time t
l_{ijt}	Distance between driver i and geohash j at time t
R_{jt}	Expected return from geohash j at time t
w	cost per distance unit
L	Maximum distance
G	Number of geohashes
d_{ijt}	Whether driver i should take the request at time t from geohash j (1 or 0)
N_t	Total number of drivers available at time t

Accomplishing this would reduce rider's waiting time and thus the total social benefit utility is higher for all the stakeholders in ride hailing business. Following is the description of the mentioned route optimization problem from the perspective of driver, rider and Uber

5.1 Driver's Problem

$$\max(R_{jt} - w * l_{ijt}) \tag{10}$$

$$\text{s.t.} \qquad l_{ijt} <= L \qquad \forall J$$

Equation 10 states that a driver wants to maximize the expected profit in facilitating a customer request as long as the distance between the driver and passenger requesting the ride is less than or equal to the maximum threshold distance that the driver can travel in an economically viable way. Basically, driver is interested in getting routed systematically by the Uber software to the most profitable locations after dropping current passengers. Driver also wants to get quick and efficient tips by the software as to where would be the high passenger demand in the coming time periods during the day. This would enable drivers to take necessary steps to be as close as possible to the potential areas of high expected passenger demand thus reaping the benefits of saving on fuel costs as well as on time resources

This would also help the drivers in increasing the total number of trips in their usual business hours and can thus achieve a higher capacity utilization rate. This means that unlike taxi drivers, Uber drivers providing ride hailing services would spend a higher proportion of their time on a trip fulfilling passenger ride request rather than having random cruising strategy in search of a passenger. Cramer and Krueger (2016) compared the capacity utilization rates between UberX and taxi services. UberX provides ride hailing services for only one customer group versus UberPool which facilitates multiple passenger requests simultaneously and thus provides a shared service for multiple customer groups with several different pickup and dropoff locations. Cramer and Krueger (2016) showed that UberX drivers have to spend less waiting time between getting a ride request and then driving down to pick up riders as compared to taxi drivers as

shown in the following figure;

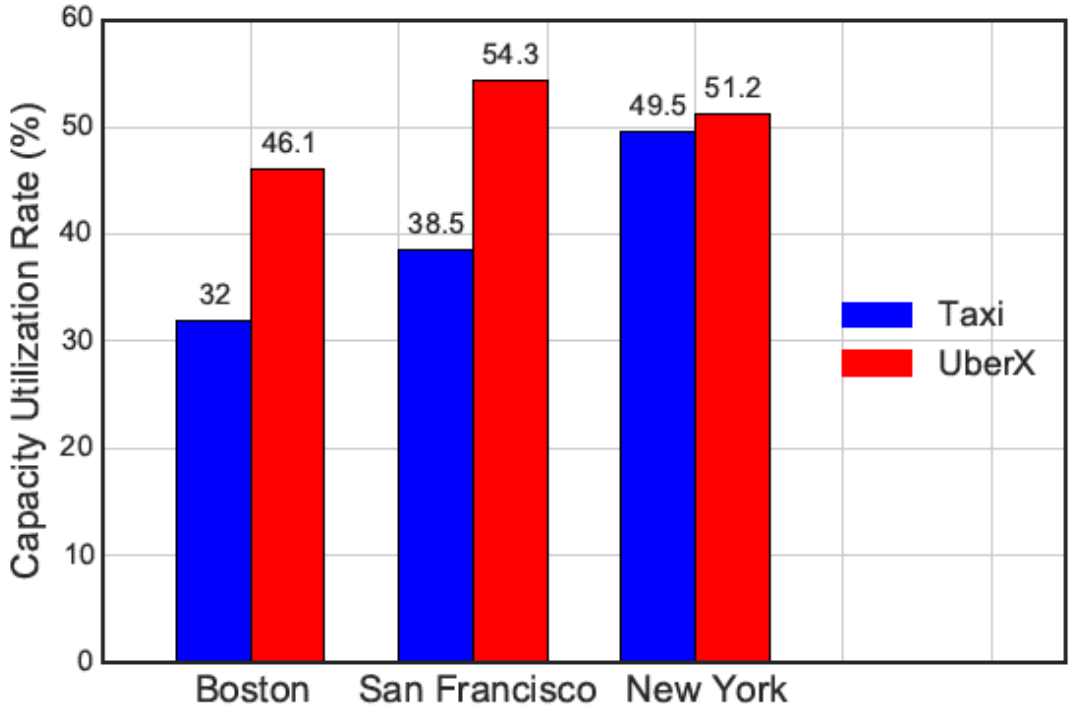

Fig. 5.1: Comparison of capacity utilization rate between UberX and taxi services (Cramer and Krueger (2016))

According to figure 5.1, capacity utilization rates for UberX is always higher than the taxi services. However, in a city like New York the difference between UberX and taxi services capacity utilization rate is not very high mainly because of a higher population density in New York city especially in downtown and Manhattan area where matching passengers with taxis is quite efficient. In cities like Boston and San Francisco where population density is smaller than in the New York city, capacity utilization rate of ride hailing services like Uber is much higher than that for taxi services. Therefore, achieving a higher capacity utilization rate is one of the top priorities for drivers providing ride hailing services. This would also enable drivers to charge less than the conventional

taxis besides earning the same revenue per hour, however due to the higher capacity utilization rate and thus higher volume of business, the total expected profit for UberX drivers is maximized.

5.2 Rider's Problem

$$\max(L - l_{ijt}) \tag{11}$$

Riders generate demand for ride hailing services on the software platform and thereby initiate the requests to be matched with the most cost effective Uber driver. For this research we have assumed that price is constant per unit of travel time and is independent of the route or passenger's pick-up point. Equation 11 states that rider's top most priority is minimum waiting time for Uber driver to pick up besides reaching the destination in the shortest possible time. Rider's other top concern is to have the ability to book Uber ride as quickly as possible after opening the Uber software application. However, as shown earlier that the riders had faced difficulty booking ride hailing services after a concert at Madison square garden which was due to the sudden spike in demand and the dynamic pricing enforced by Uber to reduce the supply and demand imbalance during that time of high demand

Therefore, the constraint here is that the rider wants to maximize the difference between the maximum distance that a driver can travel and the distance between driver i and geohash j at a certain time period t. This would ensure that the passenger gets picked up in the shortest possible time after requesting an Uber ride besides providing the passenger the quickest possible travel to the desired destination. This is particularly important in ride hailing services where each passenger's top priority is to have the

shortest travel time and yet have the lowest possible price to be paid to the Uber driver for proving the requested ride hailing service. Korolko et al. (2018) compared the rider's waiting time for the arrival of ride hailing services driver and the waiting time for the rider to catch a taxi through street hailing. The results represented in the following figure 5.2 highlight the fact that the rider's waiting time in ride hailing service is significantly less than in the conventional street taxi system. This efficiency of ride hailing platform in reducing the rider's waiting time is achieved due to number of factors including the following:

1. Ride hailing software platform's optimal allocation which uses complex mathematical models to allocate drivers to facilitate ***matching*** and thus achieve higher accuracy than taxi services street hailing except in high population density areas like New York City where taxi services and ride hailing are both competitively effective

2. Independence of drivers which does not restrict their freedom of work as much as in taxi services where drivers are bound to work for only one company at a time. However, in ride hailing riders get higher benefit and less waiting time because drivers are allowed to work for multiple companies like Uber and Lyft simultaneously

3. Higher profit margin for ride hailing service providers or drivers due to their high capacity utilization rates and less idle travelling time. This eventually also provides higher social benefit to riders who get better ride hailing services and less waiting time for both pick up and subsequent drop off to their destination address

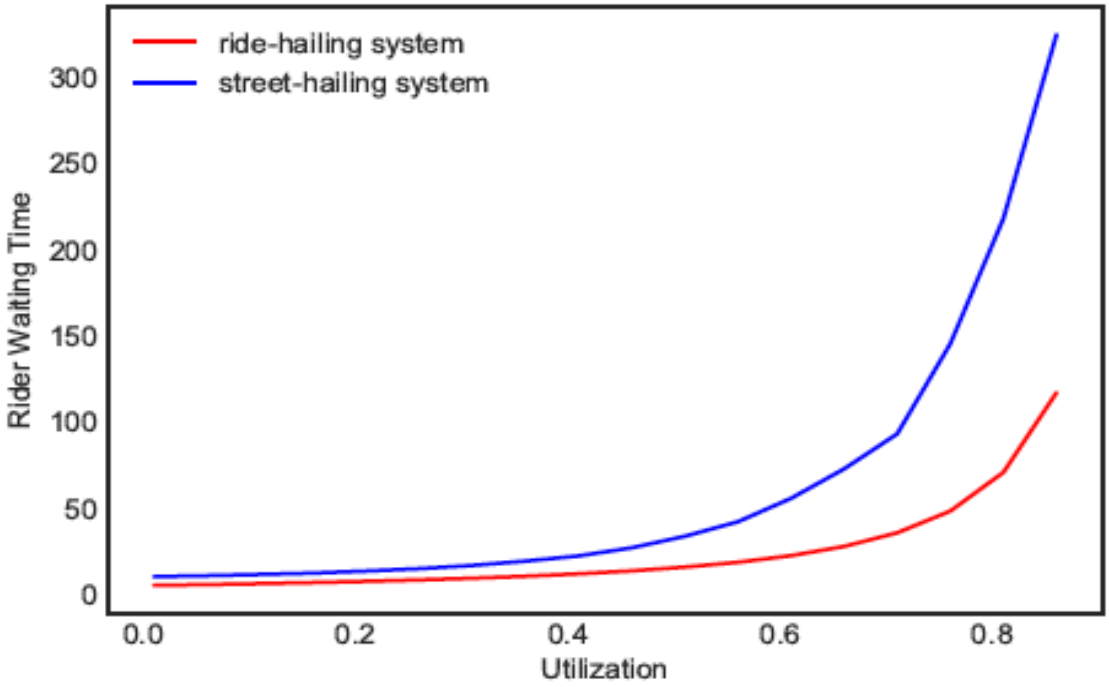

Fig. 5.2: Comparison of rider's waiting time between ride hailing and street taxi hailing services (Korolko et al. (2018))

5.3 Uber's Problem

$$\sum_{i=1}^{N_t} \sum_{j=1}^{G} (R_{jt} - w * l_{ijt}) \tag{12}$$

s.t. $$d_{ijt} <= q_{jt}$$

$$l_{ijt} <= L \qquad \forall J$$

In addition to the total cost incurred by driver and the waiting time of passenger to get picked up after requesting a ride, Uber's next concern is to maximize it's own revenue and expected profit as well. According to equation 12, Uber wants to maximize the

difference between expected revenue and cost incurred by all the drivers in navigating all the available geohashes to facilitate passenger requests. This difference is called the expected profit for all the available drivers and for the locations at a certain time period t by also keeping in sight the constraints mentioned. One of our model's defined constraints is that the total number of drivers allocated to a particular geohash should be less than or equal to the total number of expected requests in that particular geohash $\boldsymbol{d_{ijt} <= q_{jt}}$. This condition is satisfied in one of the constraints of equation 12 above

Another constraint of equation 12 is that the distance between driver i and geohash j at time t should be less than or equal to the maximum threshold distance defined earlier in our algorithm. Uber wants to ensure that it is a win-win situation for all the stakeholders involved and thus it provides the maximum utility to driver, passenger and to itself as a ride services provider. This would increase the total social benefit for all the stakeholders involved in ride hailing service. Our objective is to ensure balanced and fair driver allocation strategy so that neither the interests of drivers such as travelling time to pick passengers and the associated expected profit from the ride, nor the interests of passengers such as waiting time to get picked and the time to reach their destination are compromised

To provide drivers with high ***capacity utilization rate*** and passengers with low waiting time, Uber uses two key technology based strategies which are ***dynamic pricing*** and ***matching***. However, in our research focused on this thesis, we are only using the matching analogy in our algorithm. We are focused only on the dynamic allocation of Uber drivers to the most profitable geohashes based on the expected future passenger demand and the associated expected revenue in different time periods. Recall that each time period is of 30 minutes and we have 48 time periods in a day. Matching means the method Uber uses to dispatch drivers and allocate them to facilitate passenger

ride hailing requests in real time, whereas dynamic pricing encapsulates the pricing strategy that Uber uses to reduce the imbalance between supply and demand conditions such as a sudden spike in demand for Uber drivers that occurs after a major sporting event, a concert or during New Year's eve. Uber has particularly named dynamic pricing as surge pricing which is like an incentive strategy to compensate drivers during the periods of sudden demand surge. This high pricing method proves to be useful in reducing the rider's waiting time, providing high capacity utilization rate to drivers and also helps in minimizing the impact of drivers shortage during the periods of sudden spike in passenger demand. However, in this thesis we are assuming price to be constant per unit of time

Despite providing some immediate benefits as mentioned above, dynamic pricing has its drawbacks as well (Korolko et al. (2018)), which include price volatility and high customer turnover rate as they look for other ride hailing service providers to beat the high prices during high demand periods. Drivers also get irritated in the long term from dynamic pricing as it is often short lived and by the time they reach high demand geohash clusters, the proposed surge pricing is not effective any more. Therefore, Uber has to keep all of these constraints in close consideration while using the dynamic pricing strategy. To address this downside and to minimize price volatility due to sudden imbalance between demand and supply, Uber has recently introduced the feature called Express Pool which is based on ***dynamic waiting*** where passengers wait for the most economical ride to be connected to and in which other passengers would also be accompanying the ride. However, our research is focused only on the driver side of optimal allocation where we are allocating drivers using the predicted passenger demand and the associated expected revenue for different time periods and across various locations within the area of our designated research

Chapter 6

Solution to Driver's Problem

We start initially by dividing the area such as Bay Area, California into equally spaced tiny squares denoting ***geohashes*** and then assigning unique ids to these geohashes for easy reference. Since each city is geographically diverse in terms of usage pattern and passenger demand for ride hailing services, we need to analyze all the locations at the lowest possible granularity. Therefore we calculate expected demand and thus provide supply in terms of allocating drivers using the neighborhood clusters or geohashes in our research. We used ARIMA forecasting methods to forecast expected passenger ride hailing demand for different time periods and for each of the clusters or geohashes. This would provide easy reference to us when we would allocate drivers to potential demand locations. Also, the ids of the geohashes would keep updating whenever there is a new location that has ride hailing potential demand and thus requires corresponding supply of drivers

Using the concept of matrices and vectors for allocation of drivers,we keep changing the assigned geohash for drivers whenever there is a conflict where total number of passenger requests is less than the total number of assigned drivers. In this scenario, we allocate only the top n drivers whose expected profit from facilitating passenger requests in a particular geohash is higher and where number of drivers is equal to the number of passenger requests emerging in a certain time period. Therefore, the number of drivers whose expected profit is lower than the profit of the assigned drivers is then reassigned

to other geohashes again using their expected profits by comparing them with the profits of other drivers available to be assigned to those geohashes at a certain time period

Our methodology is inspired by Uber's current technique of managing the dispatch optimization of drivers where Uber uses geo-spatial design to divide the earth into tiny cells using the **Google S2 library**. This uses spatial indexing and the **Hilbert-space filling curve** methods to assign x and y coordinates to a point in space and thereby divides the area into equally spaced clusters or geohashes as mentioned above. **Hilbert curve** is basically a continuous fractal space-filling curve that occupies a given space and covers all the area within that space by using its curve to connect and fill the coordinates

Subsequently, we start by assigning N drivers to different $\boldsymbol{G}$ geohash locations for a time period t of 30 minutes each comprising of about 48 time periods in a 24 hours day. Secondly,we calculated $\boldsymbol{l_{ijt}}$ each driver's distance from each of the 20 different geohash clusters for the next time period t+1. Subsequently, based on each driver's current location and the point of passenger's request for a ride, we calculated expected revenue $\boldsymbol{R_{jt}}$ if a driver accepts passenger's request coming from a particular geohash and starts facilitating that request. We had earlier defined a threshold of cost per distance unit $\boldsymbol{w}$ which we then multiplied with the total distance between driver's existing location and the next passenger's location $\boldsymbol{l_{ijt}}$ to get accumulated cost per trip

Using estimated revenue, we then calculated expected profit for drivers considering their current location and the passenger requests emerging from various locations in the city. The expected profit is the difference between expected revenue and the total cost for a driver i, geohash j and at time t. For a time period t, we also calculated the expected number of passenger requests $\boldsymbol{q_{jt}}$ emerging from different geohash locations $\boldsymbol{G}$ using the expected demand for Uber drivers which we had earlier calculated using the

forecasts generated from Poisson distribution and Auto Regressive Integrated Moving Average (ARIMA) statistical models defined in chapter 3. Using expected profit, time period t and the number of passenger requests for Uber rides, we subsequently used optimization strategy in our algorithm to allocate Uber drivers to their most profitable passenger locations. We also intended to ensure that the number of drivers allocated to each geohash is always less than or equal to the total number of passenger ride requests in each location $\boldsymbol{d_{ijt} <= q_{jt}}$

By doing that we want to make sure that there is no such instant where the passenger's demand and driver's supply are not matched appropriately and thus giving rise to the supply and demand imbalance. Each passenger's ride hailing request is matched with the first available driver who is expected to have the shortest travel time to reach the passenger's pick up location using the ***Euclidean Distance*** measure to calculate straight line distances between each driver's existing location and all the available geohash locations within the metropolitan city under consideration. While allocating drivers to passenger's pick up and ride hailing requests, our most important criteria is the expected profit for the driver in case he facilitates the assigned passenger's request

Our algorithm is designed to incorporate the constraint that would check the condition if total number of customer ride hailing requests in a particular geohash at a certain time period of t=30 minutes is less than the total number of Uber drivers allocated to that specific geohash. If the above condition is met, our algorithm would then check the expected profit of each driver going to that geohash and then selects only those **n** drivers that have the highest profit. In this case, we incorporated a condition that **n** was equal to the total number of passenger requests in a particular geohash at a certain time period. Therefore, we allocated n number of drivers with the highest profits

to those geohashes where number of drivers allocated initially was greater than the total number of requests. Subsequently, we again optimized our driver allocation process to make sure that all drivers are properly allocated to their respective locations. Also, we make sure that there are no conditions where the number of allocated drivers is greater than the number of passenger requests as in that case the imbalance between supply and demand would still prevail and that we aim to minimize through this research. This constrained optimization algorithm would keep functioning until all the available drivers are allocated to their most profitable and economically feasible geohash locations and all the passenger requests emerging from different geohash locations are facilitated by assigning the closest Uber drivers given their existing location and total expected revenue in the current optimal allocation

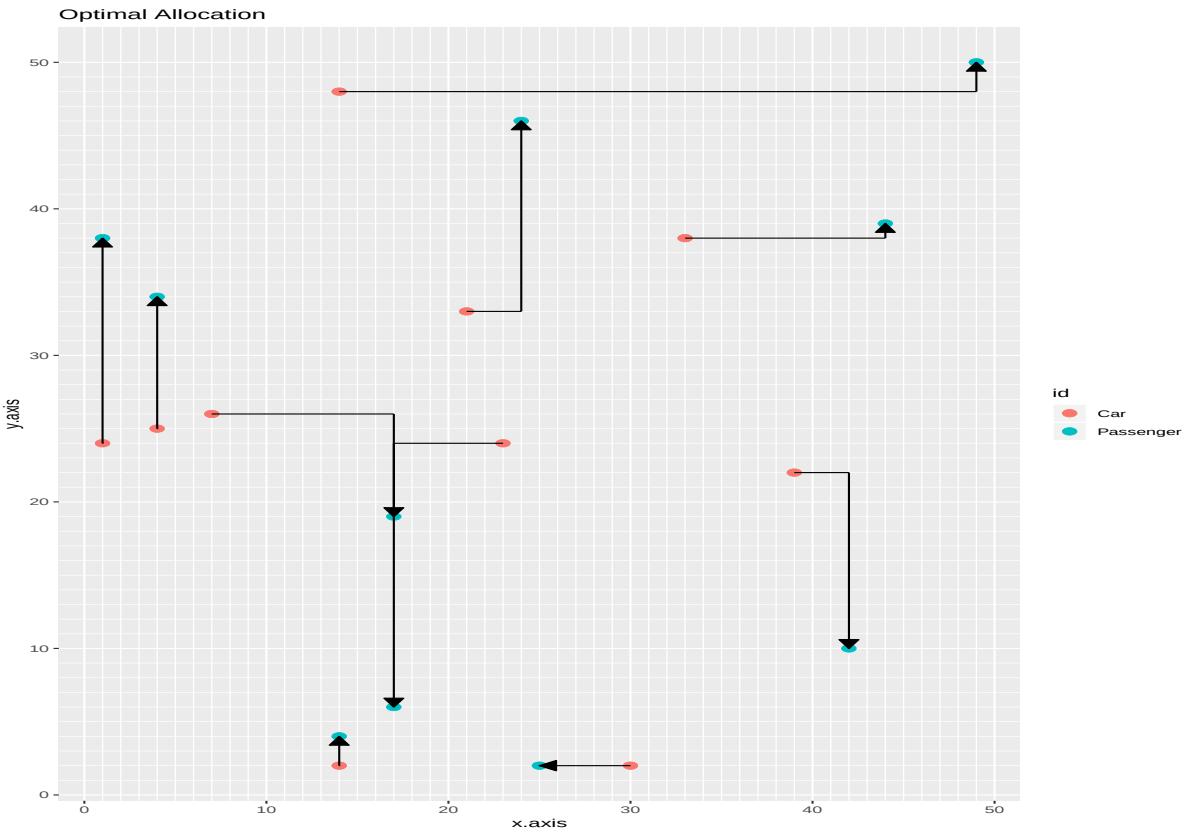

Fig. 6.1: Uber Driver Assignment

The above figure shows the optimal driver allocation after assigning them to the most profitable and economically feasible passenger location. This plot is generated in R statistical environment where we initially assigned drivers and passenger to some random locations. Subsequently we calculated distance matrix between drivers and passengers in order to assign them optimally by observing the initial position of driver and passenger as well as the final destination. We used linear programming solver in R as one of the other approaches to optimally allocate drivers to passengers ride hailing requests and there we generated the above plot. The plot shows that the driver who is nearest to the passenger's pick up request, in terms of the straight line or Euclidean distance calculated earlier, is allocated to that passenger. This Uber assignment simulation is done for a city which for our research is divided into clusters and for each cluster we calculated the expected demand using forecasting methods like Auto Regressive Integrated Moving Average (ARIMA) defined in section 3.3. Subsequently based on driver's current location and passenger's pick-up point, we calculated expected revenue for the request coming from that geohash which we then used to calculate expected profit for the drivers considering their current location and the location of a passenger. Recall from chapter 5, table 5.1 that

1. $\boldsymbol{q_{jt}}$ is the expected number of requests from geohash j at time t that we calculated using the demand predictions from Poisson distribution and ARIMA forecasting models in chapter 3

2. $\boldsymbol{l_{ijt}}$ is the distance between driver i and geohash j at time t which is then used to calculate expected profit by multiplying it with cost per distance unit $\boldsymbol{w}$

3. $\boldsymbol{R_{jt}}$ is the expected revenue from geohash j at time t

4. $\boldsymbol{d_{ijt}}$ is either 1 or 0 indicating whether the passenger request is accepted by

the nearest driver who is willing to facilitate the request

Chapter 7

Simulation of Results

The solution defined in chapter 6 is implemented in R statistical programming tool where we created representation of Uber and driver problem defined in Chapter 5 . We utilized matrix representation to simulate driver allocation across various clusters/geohashes in a city like Bay Area, California. Using sequence generation, we started by creating a regular sequence of $\boldsymbol{x}$ integers and then used random sampling to select a sample of specified size $\boldsymbol{y}$ from the elements of $\boldsymbol{x}$ where $\boldsymbol{y}$ is less than or equal to $\boldsymbol{x}$ in size. The sample of random integers drawn as above, $\boldsymbol{y}$ is supposed to be the initial location of drivers where drivers are currently located across the city. Also recall from Table 5.1 that $\boldsymbol{N_t}$ is the number of available drivers at time t,therefore the size of sample y is equal to $\boldsymbol{N_t}$

We then used matrix function to create a matrix for number of requests from geohash j at time t, $\boldsymbol{q_{jt}}$, using a sample of integers between 0 and 7 to represent number of requests and a sample size of 140. This matrix has n1 rows denoting geohashes/-clusters and n2 columns representing time periods of 30 minutes each. Similarly, for expected return from geohash j at time t, $\boldsymbol{R_{jt}}$, we used matrix formulation using a sample of integers between 10 and 100 to represent expected return and a sample size of 140. This matrix has x1 rows denoting geohashes/clusters and x2 columns representing time periods of 30 minutes each. Subsequently, we retrieved those indices in the matrix of number of requests $\boldsymbol{q_{jt}}$ where passenger requests for a particular geohash at time

t are 0 and used those indices to replace respective values in the matrix of expected return $\boldsymbol{R_{jt}}$ to 0. This is to ensure that if number of passenger request for Uber service is 0 for a geohash j and time t, then the corresponding expected revenue is also 0. To compute distance between geohashes/clusters, we used distance matrix computation to create a matrix with a distance object of integers between 1 and 20

This distance matrix is computed using **Euclidean** distance measure to compute the distances between the rows of a data matrix where the logical value indicating the diagonal and upper triangle of the distance matrix are True. To perform statistical computations on the distance matrix created above, we converted the distance matrix to a matrix object so that necessary computations could be performed. Subsequently,in order to check the indices where number of requests is greater than 0, we used conditional indexing on our number of requests $\boldsymbol{q_{ijt}}$ matrix to retrieve all the rows and one of the columns; representing a particular time period t for all the geohashes. If it is found that number of requests is greater than 0 in any geohash at a time period t , then we replaced that value with 1. The resulting object $\boldsymbol{f}$ is a vector of length equal to the number of geohashes $\boldsymbol{G}$ where we have only 1, which represents number of requests greater than 0, and 0 which denotes that number of requests are equal to 0

We then used this vector to combine by rows using the repetitions equal to the number of available drivers at time t, $\boldsymbol{N_t}$. This resulting matrix has rows equal to $\boldsymbol{N_t}$, the number of available drivers at time t, and columns equal to the number of geohashes, $\boldsymbol{G}$. Since each row in this matrix is a replication of the vector $\boldsymbol{f}$, the resultant matrix $\boldsymbol{F}$ has either 1 or 0 to represent number of requests for the available drivers at time t, $\boldsymbol{N_t}$ denoted by number of rows and total geohashes $\boldsymbol{G}$ represented by number of columns. Next, we created a vector of expected return $\boldsymbol{R_{jt}}$ for a time period t, and then used it to calculate expected profit by subtracting it from the product of total cost per distance

unit and the distance of each driver's initial or existing location from all the available geohashes, $(\boldsymbol{R_{jt} - w * l_{ijt}})$ which is our equation 10 in chapter 5.1. We then transposed the expected profit matrix calculated using equation 10 above, so that columns represent profits for each of the geohash, while rows represent each drivers existing location. In order to ensure that if number of requests $\boldsymbol{q_{jt}}$ is 0 in a certain geohash for a time period t, then the corresponding profit is also 0, we multiplied the matrix $\boldsymbol{F}$ which has all 1 and 0 defined above with the transposed matrix of the expected profit. For some drivers and locations at a time period t, we have negative profits or losses indicating that it is not economically feasible for Uber drivers to facilitate passenger requests in that geohash. So, we replaced those negative profits with 0 before doing further computations and named this new matrix as **BM**

Our next task now is to simulate the allocation of drivers to their most profitable geohashes for the next time period $\boldsymbol{t + 1}$. Using row indices and the number of rows in the matrix **BM** to denote driver numbers, we calculated the maximum of each row to represent the maximum available profit for each driver. We also calculated the corresponding geohash of every driver's maximum profit by retrieving the column name where maximum profit occurs. The resultant matrix is named **H** and it displays the driver id in column1, the next geohash they are allocated to in the next time period t+1 in column 2 and the corresponding profit of each driver for their respective geohash allocation in the mentioned time period in column 3 whereas the row indices of this matrix indicate driver's current geohash location. Recall from chapter 6 that our foremost objective is to ensure that the number of allocated drivers does not exceed the number of available passenger requests in a certain geohash at a particular time period. Therefore, after creating the matrix **H** as defined above, we calculated the frequency of each geohash after the initial driver allocation, indicating the number of drivers allocated to every geohash through the simulated allocation technique explained in this chapter.

Next, we retrieved the available number of passenger requests from the matrix $\boldsymbol{q_{jt}}$ for all the unique geohashes and for the mentioned time period. These one column matrices of frequency of each geohash, available number of passenger requests and corresponding geohash are then combined by column to form a matrix named as **mat1**

We then used a for loop on this matrix **mat1** to check for the condition in each row if at time t+1 for a certain geohash, the available number of passenger ride requests is less than the total number of drivers allocated. If the above condition is found true at any geohash then we keep appending that geohash to a vector **GA** and then use that geohash to retrieve those driver indices who have been allocated to that geohash in our earlier defined matrix **H**. Now, we use the number of available passenger ride requests from column 1 in matrix **mat1** and determine locations or indices of drivers equal to the number of available requests using their highest expected profits and remove those drivers whose expected profits were lower. We also keep concatenating these newly allocated drivers based on highest expected profits to a vector **DA**

Next, we use the vectors **GA** and **DA** to remove these allocated drivers and their respective geohash that they have to go to in the next time period t+1 to remove them from the matrix **BM** defined earlier in this chapter by replacing the profits of them with 0. After eliminating the allocated drivers and the geohashes with total number of passenger requests fully allocated with equal number of available drivers from the matrix **BM**, we again calculate the allocation of the remaining drivers to the passenger ride requests. This allocation and reevaluation of the conditions defined above are continued in the mentioned for loop until all the remaining Uber drivers are successfully allocated to the most economically feasible geohashes. This is done to ensure that the number of allocated drivers is less than or equal to the available passenger ride requests for a time period t+1 and in a specific geohash under consideration.

Chapter 8

Future Work

As a future work, we could spend more time to further improve the accuracy of our next time period forecasts generated by the various time series forecasting methods that are being used in this research. By doing that, we could also reduce the sMAPE error values calculated earlier to determine the accuracy of our results. For e.g in the ARIMA method, we could devote more resources in terms of utilizing higher computing power to choose more optimal values of parameters p, d and q so that the forecasts made by ARIMA achieve a higher degree of accuracy. The notion behind this is that if our passenger demand forecasts for ride hailing services are quite accurate then we could allocate Uber drivers to potential high demand geohashes more efficiently and thus able to reduce the demand and supply imbalance more competitively and resourcefully

For future work, we would also incorporate multiple time periods for optimal driver allocation as currently we have just used one time period. This includes generating expected passenger demand for multiple time periods instead of just one and then allocating drivers to multiple future time periods for the driver allocation optimization problem. Besides this, we would also develop an application for Uber drivers to test in real time the accuracy of our algorithm so that our model could be further improved by learning from the real results. Another thing that we would include in the future work is the concept of **surge pricing** or dynamic pricing that Uber currently uses to reduce rider's waiting time and to compensate drivers during high passenger demand periods.

We could relax the assumption that price is constant as we currently have assumed in our thesis and make price a varible that changes with demand to incorporate surge pricing in the future work. It is like an incentive strategy to compensate drivers during the periods of sudden demand surge. This high pricing method proves to be useful in reducing the rider's waiting time, providing high capacity utilization rate to drivers and also helps in minimizing the impact of drivers shortage during the periods of sudden spike in passenger demand.

Chapter 9

Conclusion

With the rise in usage of ride hailing software applications and the surge in data generated through these platforms, the demand for data analytics and machine learning algorithms to extract meaningful insights from the data has been unprecedented. Through this thesis, we strived to improve Uber driver's mobility intelligence and to provide them with better and sophisticated route recommendation systems in the form of mathematical models and algorithms that we designed in this research. We started by collecting the historical data of 41 Uber drivers who carried out ride hailing services in San Francisco, we then performed data cleaning, data wrangling and statistical analysis on this data to identify patterns,trends and relationships. We also proposed an optimization model in this thesis that would reduce the imbalance between the demand for Uber driver's ride hailing services and the corresponding supply of the drivers to match the passenger demand. This is expected to reduce the driver's idle travelling time in order to catch the next passenger request besides also decreasing the passenger waiting time for the driver to arrive at the pick-up point

Analyzing the historical data of Uber passenger rides where the drivers facilitated pick-up and drop off requests, we identified useful insights that we depicted through the visualizations prepared in R statistical tool. The insights generated thought these graphs such as average revenue per driver, daily, weekly and monthly fluctuations and demand surge patterns on holidays present useful findings, for a ride hailing company like Uber.

These findings can be used to make better business strategies especially for the future rides in order to provide the maximum benefit to the passenger as well as the driver. We also used different time-series forecasting methods like Poisson distribution, ARIMA model and Ensemble framework to predict next period demand forecasts by dividing the historical data into training and test datasets and then generating time-series passenger demand forecasts for the future periods. We also proposed that by forecasting next time period passenger demand, we can solve the driver allocation problem by assigning them to geohashes in the most optimal pattern and thus help in minimizing driver's random cruising strategies to reach high demand passenger locations.By achieving optimal driver allocation, we intended to ensure that the number of drivers allocated to a particular geohash is not greater than the number of passenger ride hailing requests in that geohash and for a certain time period of 30 minutes each

Poisson distribution and Ensemble model are two of the best performing forecasting methods in our research according to the sMAPE accuracy measure, achieving 70.08 and 70.40 error measurement respectively. We can say with confidence that our endeavor in this research work is a major contribution towards accurately predicting time-series passenger demand forecasts and thus guiding the Uber drivers towards potential high demand hotspots and therefore contribute towards minimizing the demand and supply imbalance in Uber ride hailing services. This model will be used as a recommendation system to provide smart live tips and guidance to Uber drivers about which direction they should head to after a drop off so that their idle travelling is minimum and thus capacity utilization rate is maximum.

Appendix A

R code

```
******* Preprocessing of the dataset, changing datatypes, data transformation,
creating new variables, timeperiod calculation t=30 minutes each for total
of 48 timeperiods in 24 hours ***********

> MyData <- read.csv(file="07_03_TestData.csv",stringsAsFactors = F)
> head(MyData)
> str(MyData)
> dim(MyData)
> MyData$pickup_date = as.Date(MyData$pickup_date, '%m/%d/%y')
library(lubridate)
> x=hms(MyData$pickup_time)
> MyData$Time_Period = hour(x) * 2 + ifelse((minute(x)-30)>0,2,0) +
ifelse((minute(x)-30)<0,1,0)
> MyData[order(as.Date(MyData$pickup_date, format="%m/%d/%y")),]
> library(dplyr)
MyData$geo_hash1 = str_sub(MyData$geo_hash,1,5)
length(unique(MyData[["driver_uuid"]]))
> MyData <- MyData%>%
```

```
mutate(Driver_ID = group_indices(., driver_uuid))
> days = group_by(MyData,geo_hash1, pickup_date, Time_Period)
> requests = summarize(days, Total_num_of_requests=dplyr::n())
>requests = as.data.frame(requests)
>dfnew <- requests[,c(1,2,3,4)]
>df3 = ddply(dfnew, "pickup_date", summarise, Time_Period =seq(1, 48))
>z = merge(requests,df3, all.y=TRUE)
>write.csv(z, "data1.csv", na = "0")
> df8 <- read.csv(file="data1.csv",stringsAsFactors = F,row.names=1)
> length(unique(df8[["Driver_ID"]]))
> length(unique(df8[["Time_Period"]]))
> length(unique(df8[["Total_num_of_requests"]]))
> df8$pickup_date = as.Date(df8$pickup_date, '%Y-%m-%d')
> newFrame <-aggregate(df8$Total_num_of_requests,
 list(df8$pickup_date),mean)
>  newFrame <-aggregate(df8$Total_num_of_requests,
  list(df8$pickup_date,df8$Driver_ID),mean)
>  names(newFrame) <- c("pickup_date","Driver_id",
  "Total_num_of_requests")
> ggplot(newFrame, aes(x = newFrame$Group.1, y = newFrame$x)) +
 geom_point(size = 2,shape=19)
> ggplot(newFrame, aes(month, x)) + geom_line()
> bp= ggplot(data=newFrame, aes(x=factor(month),y=x))
> bp= ggplot(data=newtable, aes(x=factor(Driver_ID),y=avg))
> bp+geom_boxplot(fill='white',color="darkred")+xlab("Driver ID") +
 ylab("Total Revenue")+  theme(text = element_text(size=15))
```

```
> newFrame<- newFrame %>%mutate(year = year(Group.1))
>newFrame$day = strftime(newFrame$Group.1,'%A')
>newFrame$month = strftime(newFrame$Group.1,'%m')

*********Average Revenue for Geohashes*************

>df_freq=data.frame(table(MyData$geo_hash))
>df_aggr=aggregate(MyData$total, by=list(geo_hash=MyData$geo_hash),
FUN=sum)
>df_aggr$freq=df_freq$Freq
>df_aggr$avg=df_aggr[,2] / df_aggr[,3]
>min(df_aggr$avg)
>max(df_aggr$avg)
>newtable <- merge(MyData,df_aggr, by  = "geo_hash")

>ggplot(drier_freq, aes(x = Var1, y = Freq)) +
geom_col(fill='blue',color="black")+xlab("Driver ID") +
ylab("Number of Trips")+  theme(text = element_text(size=15))

***********Average Revenue for Drivers*********

> df_freq2=data.frame(table(MyData$Driver_ID))
> df_aggr2=aggregate(MyData$total, by=list(Driver_ID=MyData$Driver_ID),
FUN=sum)
> df_aggr2$freq=df_freq2$Freq
> df_aggr2$avg=df_aggr2[,2] / df_aggr2[,3]
```

```
> ggplot(df_aggr2, aes(x = Driver_ID, y = avg)) +
geom_col(fill='yellow',color="black")+xlab("Driver ID") +
ylab("Average Revenue")+  theme(text = element_text(size=15),
panel.grid.major = element_blank(),
panel.grid.minor = element_blank(),
panel.background =
element_rect(fill = "transparent",colour = NA),
plot.background =
element_rect(fill = "transparent",colour = NA))+
scale_x_continuous(breaks=seq(1,41,1))+
scale_y_continuous(breaks=seq(0,100,10))

**********Visualizations using ggplot package*********

> requets_month <- newFrame %>%
     group_by(month, year) %>%
     summarise(mean_requests = mean(x))
> requets_month %>%
     ggplot(aes(x = month, y = mean_requests)) +
     geom_bar(stat = "identity", fill = "darkorchid4") +
     facet_wrap(~ year, ncol=3) +
     labs(title = "Mean Monthly Requests",
          subtitle = "Data plotted by year",
          y = "Daily Total no.of Requests",
          x = "Month") + theme_bw(base_size = 15)
> requets_day <- newFrame %>%
```

```
          group_by(Group.2, day) %>%
          summarise(means_requests = mean(x))
> requets_day %>%
          ggplot(aes(x = Group.2, y = means_requests)) +
          geom_point(stat = "identity", fill = "darkorchid4") +
          facet_wrap(~ day, ncol = 3) +
          labs(title = "Mean Daily Requests",
                         subtitle = "Data plotted by day",
                         y = "Mean Daily Requests",
                         x = "Driver_id") + theme_bw(base_size = 15)
>newFrame %>%
ggplot(aes(x = Group.2, y = x)) +
      geom_bar(stat = "identity",color = "darkorchid4") +
      facet_wrap( ~ month, ncol = 3) +
      labs(title = "Monthly Total Request - Driver_id",
           subtitle = "Data plotted by month",
           y = "Total Requests",
           x = "Driver_id") + theme_bw(base_size = 15)


hist(newFrame$Driver_id)
library(plyr)
library(dplyr)
geohash_vector <- pull(MyData, geo_hash1)
u = unique(geohash_vector)
MyData1 <- arrange(MyData,pickup_date)
> values = seq(from = as.Date("2014-05-05"), to = as.Date("2017-03-07"),
```

```
by = 'day')
> for(i in 1:length(u)){geohash <- u[i]
 mini.df <- cbind(data.frame(c(values)))
 mini.df = ddply(mini.df, "values", summarise, Time_Periods =seq(1,48))
 mini.df = cbind(mini.df, geohash)
 assign(paste("mini.df", i, sep="."), mini.df)
 }
> a=filter(z, geo_hash1== '9q8z')
> unique(MyData$geo_hash1)
> table(unlist(MyData$geo_hash1))
> colnames(mini.df.1)[colnames(mini.df.1)=="values"] <- "pickup_date"
colnames(mini.df.1)[colnames(mini.df.1)=="Time_Periods"] <- "Time_Period"
> x= left_join(mini.df.1,a)
> x$geo_hash1=NULL
> write.csv(x, "mini.df.1.csv", na = "0")

**********Data Transformation*********

> days1 = group_by(MyData2,geo_hash1, pickup_date, Time_Period)
> requests1 = summarize(days1, Total_num_of_requests=dplyr::n(),
Total_num_of_drivers=n_distinct(Driver_ID))
> requests1 = as.data.frame(requests1)
> df_1 <- requests1[,c(1,2,3,5,6)]
> df_0 = ddply(df_1, .(geo_hash1,pickup_date), plyr::summarise,
Time_Period =seq(1, 48))
> z1 = merge(df_1,df_0,all.y=TRUE)
```

```
> write.csv(z1, "data_1.csv", na = "0")
> df_8 <- read.csv(file="data_1.csv",stringsAsFactors = F,row.names=1)

**********Creating new variables Day,Month & Week*********

> df_8$Day = strftime(df_8$pickup_date,'%a')
> df_8$Day_month = strftime(df_8$pickup_date,'%d')
> df_8$Month = strftime(df_8$pickup_date,'%b')
> df_8$Week_of_Year = strftime(df_8$pickup_date,'%V')
> geohash1 = filter(df_8, geo_hash1=="9q8uv")
> df_8 <- df_8 %>% mutate_if(is.character,as.factor)
<<<<geocodes <- geocode(as.character(MyData2$dropoff_address))

**********Changing datatypes*********

> df_8$Day <- factor(df_8$Day, levels=c("Mon", "Tue", "Wed", "Thu","Fri","Sat",
"Sun"))
> df_8$Month <- factor(df_8$Month, levels=c("Jan", "Feb", "Mar", "Apr","May","
Jun","Jul","Aug","Sep","Oct","Nov","Dec"))
> requests_day <- df_8 %>%
     group_by(Day) %>%
     summarise(means_requests = sum(Total_num_of_requests)/
     sum(Total_num_of_drivers))
> requests_day %>%
     ggplot(aes(x = Day, y = means_requests,group=1)) +
     geom_line() +geom_point(color = "darkorchid4")+
```

```
     labs(title = "Mean Daily Requests",
          subtitle = "Data plotted by day",
          y = "Tot.num of Requests/Tot.num of Drivers ",
          x = "Day") + theme_bw(base_size = 15)
> requests_month1 <- df_8 %>%
     group_by(Day_month,Month) %>%
     summarise(mean_requests = mean(Total_num_of_requests))
> requests_month1 %>%
     ggplot(aes(x = Day_month, y = mean_requests,group=1)) +
     geom_line(color = "darkorchid4") +geom_point(color = "darkorchid4")+
     facet_wrap(~ Month, ncol=3) +
     labs(title = "Mean Daily Requests",
          subtitle = "Data plotted by day",
         y = "Daily Total no.of Requests/Tot.num.of drivers",
          x = "Day_month") + theme_bw(base_size = 15)+
          theme_bw() + theme(axis.text.x = element_text(colour = "grey20",
          size = 10, angle = 90, hjust = 0.35,
          vjust = 0.35),
        axis.text.y = element_text(colour = "grey20", size = 12),
        text = element_text(size = 16))

>requests_timeperiod <- df_8 %>%
     group_by(Time_Period) %>%
     summarise(mean_requests = sum(Total_num_of_requests)/
     sum(Total_num_of_drivers))
> requests_timeperiod %>%
```

```
     ggplot(aes(x = Time_Period, y = mean_requests,group=1)) +
     geom_line(color = "darkorchid4") +
     labs(title = "Mean Time_Period Requests",
          subtitle = "Data plotted for each time period",
          y = "Daily Total no.of Requests/
          Tot.num.of drivers",
          x = "Time_Period") + theme_bw(base_size = 15)

> requests_timeperiod <- df_8 %>%
     group_by(Time_Period,Day) %>%
     summarise(mean_requests = sum(Total_num_of_requests)/
     sum(Total_num_of_drivers))

> requests_timeperiod %>%
     ggplot(aes(x = Time_Period, y = mean_requests,group=1)) +
     geom_line(color = "darkorchid4") +geom_point(color = "darkorchid4")+
     facet_wrap(~ Day, ncol=3) +
     labs(title = "Mean Time_Period Requests by Day",
          subtitle = "Data plotted by day",
         y = "Daily Total no.of Requests/Tot.num.of drivers",
          x = "Time_Period") + theme_bw(base_size = 15)+
          theme_bw() +
    theme(axis.text.x = element_text(colour = "grey20", size = 10, angle = 90,
    hjust = 0.35, vjust = 0.35),
    axis.text.y = element_text(colour = "grey20", size = 12),
          text = element_text(size = 19))
```

```
> requests_timeperiod %>%
     ggplot(aes(x = Time_Period, y = mean_requests,group=1)) +
     geom_line(color = "darkorchid4") +
     labs(title = "Mean Time_Period Requests",
          subtitle = "Data plotted for each time period",
          y = "Daily Total no.of Requests/Tot.num.of drivers",
          x = "Time_Period") + theme_bw(base_size = 15)

>requests_week_of_year <- df_8 %>%
     group_by(Week_of_Year) %>%
     summarise(means_requests = sum(Total_num_of_requests)/
     sum(Total_num_of_drivers))

>requests_week_of_year %>%
     ggplot(aes(x = Week_of_Year, y = means_requests,group=1)) +
     geom_line() +geom_point(color = "darkorchid4")+
     labs(title = "Mean week_of_year_Requests",
          subtitle = "Data plotted by week of year",
          y = "Tot.num of Requests/Tot.num of Drivers ",
          x = "Week_of_Year") + theme_bw(base_size = 15)

> df_8$day_Month=as.integer(df_8$day_Month)
> df_8$week_of_Month=ceiling((df_8$day_Month)/7)
> df_8 <- df_8 %>% mutate_if(is.numeric,as.factor)
> df_8$Total_num_of_requests=as.integer(df_8$Total_num_of_requests)
```

```
> df_8$Total_num_of_drivers=as.integer(df_8$Total_num_of_drivers)

**********Driver Allocation Problem*********
****** Creating matrices for number of requests, expected return and distance***

> no.ofrequests<-matrix(sample(0:4,140,replace=TRUE),ncol=7,byrow=FALSE)
> exp.return<-matrix(sample(10:100,140,replace=TRUE),ncol=7,byrow=FALSE)
> exp.return[which(no.ofrequests == 0)] <- 0
> driver_reqst=matrix(data = 0:1, nrow = 10, ncol = 20)
> vreqs=no.ofrequests[1:20,1]

N=20
G=20
x=1:20
> Dist_matrix=dist(x,method="euclidean",diag=TRUE,upper = TRUE)

***Randomly Allocate drivers to geohashes***:
***vector of size 1:N, elements of this vector are an integer b/w 1:G***
**** N= number of drivers, G= number of geohashes ******

> x1=seq(1,20)
>dl:driver location
dl=sample(x1,N)
library(disto)
# create a disto connection
> dio <- disto(objectname = "Dist_matrix")
```

```
unclass(dio)

summary(dio)

library(robustbase)

> # quick plots

> plot(dio, type = "dendrogram")

> Ljt=dio[1:20,1]

Ljt

> vRjt=exp.return[1:20,1] (for t=1)

vRjt

sum=0

Sum=0

w=15

> for (i in 1:N){

 sum=sum(vRjt-(w*Ljt))

 Sum=Sum+sum

 sum=0

 print(Sum)}

 Sum

>dd=as.matrix(Dist_matrix)

> dd[,dl[2]]

> dd[dl[2],]

> for (i in 1:N){

 dm=dd[,dl[i]]

 dl2[i]=which.max(vRjt-(w*dm))}

> vreqs[(sort(unique(H[,2])))]<x1[,1]

> driver_reqst=matrix(data = 0:1, nrow = 10, ncol = 20)
```

```
> H=cbind(1:nrow(t(vRjt-(w * dd[,dl]))), max.col(t(vRjt-(w*dd[,dl])),'first'))
> for(i in 1:nrow(mat1)){
     if(mat1[i,][1]>mat1[i,][2]){
         print(mat1[i,][1])
     }
```

**** Retrieving only the passenger requests for timeperiod t=1 and assign 1 if requests are greater than 0 in all the geohashes ***

***** Calculating expected profit for all the drivers if requests are greater than 0****

*** Calculating a frequency table to show the number of allocated drivers and the available total number of requests in all the geohashes****

```
>f=no.ofrequests[,1]
>f1=no.ofrequests[,1]
>f[f>0]<-1
>F<-rbind(f)[rep(1,N),]
>BM=t(vRjt-(w*dd[,dl])) * F
>BM[BM<0]<-0
>H=cbind(1:nrow(t(vRjt-(w*dd[,dl]))), max.col(BM),apply(BM,1,max),
sample(1:G,N,replace=T))
>H[H[,3]!=0,]
BMnew<-BM
> Freq=table(H[,2])
> freq=as.matrix(Freq)
> H1=as.matrix(H)
```

```
> vreqs=no.ofrequests[1:20,1]
> unique(vreqs[H[,2]])
> vreqs[(sort(unique(H[,2])))]
> vreqs[(sort(unique(H[,2])))]>=freq[,1]
> mat1 <- cbind(vreqs[(sort(unique(H[,2])))], freq,  sort(unique(H[,2])))
> which.maxn(H[,2]==mat1[7,][3],n=mat1[6,][1])
> sort(H[,3][which.maxn(H[,2]==18,n=3)],decreasing =TRUE)[1:2]
> which.maxn(H[,2]==18,n=3)
> H[,3][which.maxn(H[,2]==18,n=3)]
```

changing the column names

```
> colnames(mat1)=c("Total no.of requests","Total no.of drivers",
"Geohash/Location")
> colnames(H)=c("Driver id","Geohash/Location","Expected Profit")
> colnames(mat1)=NULL
> colnames(H)=NULL
```

****Driver Allocation by checking if number of requests is less than the number of allocated drivers, then reassigning the allocation for all geohashes****

Appending the allocated drivers and geohashes to empty vectors for easy reference *

```
> h=hash()
> x <- list()
```

```
> i=1
> y=1
> DA=c()
> GA=c()
> for(i in 1:nrow(mat1)){
               if(mat1[i,][1]<mat1[i,][2]){
                           x1=(mat1[i,][3])
                           cat("Geohash: ", x1,"\n")
                           GA=append(GA,x1)
                           aux=which(H[,2]==mat1[i,][3])
                           aux=aux[which.maxn(H[aux,3],n=mat1[i,][1])]

    for(y in 1:length(aux)){
         cat ("Allocated Driver number",y,"is:", aux[[y]], "\n")
            DA=append(DA,aux[[y]])}
#           my_list= list(DA,GA)
#           my_list_of_lists <- append(my_list_of_lists,list(my_list))
#           x=hash(keys=GA,values=DA)
#           h = hash().set(h, keys=GA, values=DA)
            y=list("Geohashes" = GA, "Allocated Drivers"=DA)

            x=append(x,y)
#   DA=append(which.maxn(H[,2]==mat1[i,][3],n=mat1[i,][1]))
            BMnew[DA,]<-0
            BMnew[,GA]<-0
            Ht1=cbind(1:nrow(t(vRjt-(w*dd[,dl]))), max.col(BMnew),
```

```
            apply(BMnew,1,max),
            sample(1:G,N,replace=T))
#           print(Ht1)
            Ht1[Ht1[,3]==0,2]=0
#           Ht1[Ht1[,3]==0,4]=0
            Ht1[Ht1[,3]!=0,]
            Ht1[auxe1,2]=mat1[i,][3]
            Ht1[auxe1,3]=H[auxe1,3]



               }
}
```

```
> for(i in 1:nrow(mat1)){
    if(mat1[i,][1]<mat1[i,][2]){
          auxe=which(H[,2]==mat1[i,][3])
          auxe1=auxe[which.maxn(H[auxe,3],n=mat1[i,][1])]
          Ht1[auxe1,2]=mat1[i,][3]
          Ht1[auxe1,3]=H[auxe1,3]


               }
}
```

```
> x=as.data.frame(Ht1)
> coutx=x %>% group_by(x[,2]) %>% summarize(Count=n())
> countx=as.matrix(coutx)
```

```
countx=cbind(countx,vreqs[countx[,1]])
> DA1=c()
> GA1=c()
> for(i in 1:nrow(countx)){
          if(countx[i,][2]>countx[i,][3]){
          x2=(countx[i,][1])
          GA1=append(GA1,x2)
          aux1=which(Ht1[,2]==countx[i,][1])
          aux1=aux1[which.maxn(Ht1[aux1,3],n=countx[i,][3])]
          for(y in 1:length(aux1)){
          DA1=append(DA1,aux1[[y]])}
          BMnew[,GA1]<-0
          BMnew[DA1,]<-0
          Ht1a=cbind(1:nrow(t(vRjt-(w*dd[,dl]))), max.col(BMnew),
          apply(BMnew,1,max),sample(1:G,N,replace=T))
          Ht1a[Ht1a[,3]==0,2]=0
          Ht1a[Ht1a[,3]==0,4]=0
     }
 }
```

Driver Allocation when considering next 2 time periods for the expected number of passenger's ride hailing requests*

```
>t2=c()
> for(i in 1:ncol(no.ofrequests[,1:2])){
                f=no.ofrequests[,i]
```

```
                f[f>0]<-1
                F<-rbind(f)[rep(1,N),]
                BM2=t(exp.return[,i]-(w*dd[,dl])) * F
                BM2[BM2<0]<-0
      H2=cbind(1:nrow(t(exp.return[,i]-(w*dd[,dl]))),max.col(BM2),
      apply(BM2,1,max),sample(1:G,N,replace=T))
                t2=cbind(t2,H2)
}
>vreqs1=no.ofrequests[,1:2]
>Freq1=table(t2[,2])
>freq1=as.matrix(Freq1)
>Freq2=table(t2[,6])
>freq2=as.matrix(Freq2)
>library(dplyr)
>ll <- list(vreqs1[,1][(sort(unique(t2[,2])))],freq1,sort(unique(t2[,2])),
vreqs1[,2][(sort(unique(t2[,6])))],
freq2,sort(unique(t2[,6])))
>mat2=lapply(ll, function(x) x[1: max(sapply(ll, length))]) %>% do.call(cbind, .)
>mat2[is.na(mat2)] <- 0

***** Incorporating the rider's waiting time cost to determine how allocations
and expected profit change *********

>dd[cbind(H[,2],H[,4])] ***** to calculate distance between passenger's pickup
and drop off ***
>DB1=c()
```

```
>DB2=c()


>Dest=sample(1:G,N,replace=T)
>for (x in c(0:10)){
      BM=t(vRjt-((w+x)*dd[,dl])) * F
      BM[BM<0]<-0
      H=cbind(1:nrow(t(vRjt-((w+x)*dd[,dl]))), max.col(BM),apply(BM,1,max),Dest)
      H[H[,3]!=0,]
      BMnew<-BM
      Freq=table(H[,2])
      freq=as.matrix(Freq)
      mat1 <- cbind(vreqs[(sort(unique(H[,2])))], freq, sort(unique(H[,2])))
      print(mat1)
      print(H)
      DB=append(DB,sum(H[,3]+x*dd[cbind(H[,2],H[,4])]))
 }


>  BM2=t(-((w)*dd[,dl])) * F
#      BM2[BM2<0]<-0
      H2=cbind(1:nrow(t(-((w)*dd[,dl]))), max.col(BM2),vRjt[],Dest)
      H2[H2[,3]!=0,]
      BM2new<-BM2
      Freq2=table(H2[,2])
      freq2=as.matrix(Freq2)
      mat2 <- cbind(vreqs[(sort(unique(H2[,2])))], freq2, sort(unique(H2[,2])))
      print(mat2)
```

```
        print(H2)
        DB2=sum(H2[,3])

         BM=t(vRjt-((w+x)*dd[,dl])) * F
        BM[BM<0]<-0
        H=cbind(1:nrow(t(vRjt-((w)*dd[,dl]))), max.col(BM),apply(BM,1,max),Dest)
        H[H[,3]!=0,]
        BMnew<-BM
        Freq=table(H[,2])
        freq=as.matrix(Freq)
        mat1 <- cbind(vreqs[(sort(unique(H[,2])))], freq, sort(unique(H[,2])))
        print(mat1)
        print(H)
        DB1=sum(H[,3])

> DB=c()
> DB2=c()
> Dest=sample(1:G,N,replace=T)
> for (x in c(0:3)){
            BM=t(vRjt-((w+x)*dd[,dl])) * F
            BM[BM<0]<-0
            H=cbind(1:nrow(t(vRjt-((w+x)*dd[,dl]))), max.col(BM),
            apply(BM,1,max),Dest)
            H[H[,3]!=0,]
            print (x)
            print(H)
```

```
DB1=sum(H[,3])
print(DB1)
BMnew<-BM
Freq=table(H[,2])
freq=as.matrix(Freq)
mat1 <- cbind(vreqs[(sort(unique(H[,2])))], freq, sort(unique(H[,2])))
print(mat1)
for(i in 1:nrow(mat1)){
        if(mat1[i,][1]<mat1[i,][2]){
                x1=(mat1[i,][3])
                cat("Geohash: ", x1,"\n")
                GA=append(GA,x1)
                aux=which(H[,2]==mat1[i,][3])
                aux=aux[which.maxn(H[aux,3],n=mat1[i,][1])]

       for(y in 1:length(aux)){
              cat ("Allocated Driver number",y,"is:", aux[[y]], "\n")
                            DA=append(DA,aux[[y]])}
                BMnew[DA,]<-0
                BMnew[,GA]<-0
      Ht1=cbind(1:nrow(t(vRjt-((w+x)*dd[,dl]))), max.col(BMnew),
      apply(BMnew,1,max),Dest)
#                       Ht1[Ht1[,3]==0,2]=0
#                       Ht1[Ht1[,3]!=0,]
                        Ht1[aux,2]=mat1[i,][3]
                        Ht1[aux,3]=H[aux,3]
```

```
                        length(which(Ht1[,3]==0))
          Ht1[Ht1[,3]==0,3]=sample(10:100,length(which(Ht1[,3]==0)),replace=T)
                        print(Ht1)

        }
    }
    DB2=append(DB2,sum(Ht1[,3]))
    DB=append(DB,sum(Ht1[,3]+x*dd[cbind(dl,Ht1[,2])]))
   x=(vRjt-((w)*dd[,dl]))
   x[x<0]=0
   print(x)
 }
```

```
******* Uber driver allocation to passenger requests
using lpsolve in R *****
***https://www.r-bloggers.com/uber-assignment-with-lpsolve/*****
*****creating passenger's initial and final locations and driver's
initial position****
```

```
 create_passenger = function(id){

    initial.position = sample(50, 2, replace = TRUE)
    final.destination = sample(50, 2, replace = TRUE)

    return(list('number' = id, 'initial' = initial.position,
                'final' = final.destination))
```

```
}
create_car = function(id){

    initial.position = sample(50, 2, replace = TRUE)

    return(list('number' = id, 'position' = initial.position))
}
distance = function(x,y){
    sum(abs(x-y))
}
distance.matrix = function(cars, passengers){

    d.matrix = matrix(0, nrow = length(cars), ncol = length(cars))

for (i in 1:length(cars)){
    for (j in 1:length(passengers)){
       d.matrix[i,j] = distance(cars[[i]]$position, passengers[[j]]$initial)
        }
    }
    return(d.matrix)

}
passengers = lapply(seq(1:10), create_passenger)
cars = lapply(seq(1:10), create_car)
d.matrix = distance.matrix(cars, passengers)
opt.allocation = lp.assign(d.matrix)
```

```
passengers.points = sapply(passengers, function(x) x$initial)
cars.points = sapply(cars, function(x) x$position)
points = t(cbind(passengers.points, cars.points))
assignments = apply(opt.allocation$solution, 1, which.max)
#checking the assignment for each car
df1 = data.frame('x.axis' = points[,1],
                 'y.axis' = points[,2],
                 'id' = c(rep('Passenger',10), rep('Car',10)))
df.assign1 = data.frame('x' = cars.points[1,],
                        'y' = cars.points[2,],
                        'xend' = passengers.points[1,assignments],
                        'yend' = cars.points[2,])
df.assign2 = data.frame('x' = passengers.points[1,assignments],
                        'y' = cars.points[2,],
                        'xend' = passengers.points[1,assignments],
                        'yend' = passengers.points[2,assignments])
ggplot(df1, aes(x.axis,y.axis)) + geom_point(aes(color = id, group = id),
size = 3) +
geom_segment(aes(x = x, y = y, xend = xend, yend = yend), data = df.assign1) +
geom_segment(aes(x = x, y = y, xend = xend, yend = yend), data = df.assign2,
                 arrow = arrow(length = unit(0.02, "npc"), type = 'closed')) +
    scale_x_continuous(minor_breaks = seq(1, 50, 1)) +
    scale_y_continuous(minor_breaks = seq(1, 50, 1)) +
        ggtitle('Optimal Allocation')
```

Monte Carlo Simulation:(Passenger Waiting Time)*

```
simulations = function(N, MC) {

     ncars = N
     times = matrix(0, nrow = MC, ncol = N)

     for (i in 1:MC){
         passengers = lapply(seq(1:10), create_passenger)
         cars = lapply(seq(1:ncars), create_car)

         d.matrix = distance.matrix(cars, passengers)
         opt.allocation = lp.assign(d.matrix)
         times[i,] = colSums(opt.allocation$solution*opt.allocation$costs)
         # waiting time for each passenger
     }
     return(times)
 }
results = lapply(seq(10,30,2), simulations,
MC = 500) # MC = 500 just to save some time

df2 = data.frame('WaitingTime' = sapply(results, mean),
                 'LB' = sapply(results, quantile, probs = 0.10),
                 'UB' = sapply(results, quantile, probs = 0.90),
                 'Cars' = seq(10,30,2))

> ggplot(df2, aes(x = Cars)) + geom_line(aes(y = WaitingTime), lwd = 1.2) +
```

```
geom_ribbon(aes(ymin = LB, ymax = UB), fill = 'blue', alpha = .3)
```

**** Below code Moreira-Matias et al. (2012) is used for forecasting expected Passenger demand for Uber ride hailing services using which we would allocate drivers****

```
t1 <- as.POSIXct("2014-05-05")
t2 <- as.POSIXct("2017-03-07")
t_n_w = floor (as.double(difftime(t2,t1,unit="weeks")))
train_w = floor(t_n_w*2/3)
test_w = t_n_w-train_w
train_r = train_w*7*48
test_r = test_w*7*48

Train set :  98 weeks =98*7*48=32928: Vector!!!
Test set :   51 weeks =50*7*48=16800: Vector!!!

df4 =read.csv('mini.df.1.csv')
df4$X=NULL

#setwd(path)

getwd()
library(forecast)

#write.csv(data_train, 'train.csv')

#data=read.csv('data_s.csv')
```

```
data=df4$Total_num_of_requests


calsmp=function(n,pred, rd)
{
  s=sum(abs(( pred-rd))/( pred+rd+1))/n
  return(s)
}


demandtable=function(x,li){
  rownames(li[[x]]) <- weekdays(as.Date(4,"1970-01-01",tz="GMT")+0:6)
  colnames(li[[x]])<-colnames(li[[x]], do.NULL = FALSE, prefix = "Period")
  as.table(li[[x]])
}
table_pr=function(m){
  rownames(m) <- weekdays(as.Date(4,"1970-01-01",tz="GMT")+0:6)
  colnames(m)<-colnames(m, do.NULL = FALSE, prefix = "Period")
  m=as.table(m)
  return(m)
}


calculsum=function(z,np,n,liste){
  sapply(1:np,function(y) sum(sapply(1:n, function(x)liste[[x]][z,y])))
}



#Function returning predictions using Time-varying Poisson Model
```

```
#Input: n1: number of weeks considered in the train set
#       n2:number of weeks considered in the test set
#       n:number of periods in the day ( our case study 48 periods)

#       list_p: List of the historical demand stored
#       in tables(rows: day of the week/ columns: day Period
#(our case P=30-->48 period))

#       list_a: List of the demand of the test set stored
#       in tables(rows: day of the week/ columns: day Period
(our case P=30-->48 period))
#       Same order of days  as list_p

#Output: vector of the predicted number of services

poisson_pred=function(n,n1,n2,list_p, list_a){
  sum_past=lapply(1:7,function(x) calculsum(x,n,n1,list_p))
  sum_past <- table_pr(matrix(unlist(sum_past), ncol =n, byrow = TRUE))
  p_1=round(sum_past/nbr_week)

  #updating the set of the historical demand for each week for the test set
  sum_past_list=lapply(1:(n2-1),function(x) sum_past-list_p[[x]]+list_a[[x]])

  poisson_list=lapply(1:n2,function(x) if(x==1){p_1}else
  {round(sum_past_list[[x-1]]/nbr_week)})
```

```
  pred_p=unlist(lapply(1:n2, function(x) as.vector(t(poisson_list[[x]]))))
  return(pred_p)
}

calculwsum=function(n,z,num,w,l){
  sapply(1:n, function(y) round(sum(sapply(1:num, function(x)l[[x]][z,y]
  *w[num-x+1]))/sum(w[1:num]))
  )
}

#Size of Memory for W.Poisson Model
#Function returning predictions using Time-varying Poisson Model
#Input: n1: number of weeks considered in the train set
#       n2:number of weeks considered in the test set
#       n:number of periods considred in one day (48 our case)
#       list_p: List of the historical demand stored
#       in tables(rows: day of the week/ columns: day Period
#(our case P=30-->48 period))
#       list_a: List of the demand of the test set stored
#       in tables(rows: day of the week/ columns: day Period
#(our case P=30-->48 period))
#       Same order of days  as list_p
#       w/num= size of memory considered for W.POISSON Model
#       Consult the paper Part III.B eq(5)
#Example
alpha=0.4  #user_defined variable
```

```
we= sapply(1:20, function(y)alpha*(1-alpha)^(y-1))
num=max(which(we> 0.01))

#Output: vector of the predicted number of services

wpoisson_pred=function(n,n1,n2,num,w,list_p,list_a){
  list_p2=lapply(1: num, function(x) list_p[[n1-num+x]])
  pdemand_list=append(list_p2,list_a)

  #updating the set of the historical demand for each week for the test set

  up_pdemand_list=lapply(1:n2, function(x)pdemand_list[x:(num-1+x)])
  wpoisson_list=lapply(1:n2, function(y) table_pr(matrix(unlist(lapply(1:7,
  we function(x) calculwsum (n,x,num,w,up_pdemand_list[[y]]))),
  ncol = n, byrow = TRUE)))
  pred_wp=unlist(lapply(1:n2, function(y) as.vector(t(wpoisson_list[[y]]))))
  return(pred_wp)

}



calculaModeloArima<-function(timeseries,th=14*48)
{
  #if th<15 days, produce warning
```

```
if (length(timeseries)<th)
{
  print(length(timeseries))
  print(th)
  print("WARNING: The supplied series has a size smaller than expected !!!")
}

#Select the model
fit <- tryCatch(auto.arima(timeseries,allowdrift=FALSE,seasonal = FALSE),
error=function(e) e)
if (is.list(fit))
{
  arma<-fit$arma
  myOrder<-c(arma[1],arma[6],arma[2])
  if (is.vector(myOrder) && length(myOrder)==3)
  {
    if ((myOrder[1]==4 && myOrder[2]==0 && myOrder[3]==4) ||
    (myOrder[1]==0 && myOrder[2]==0 && myOrder[3]==0) ||
    (myOrder[1]==4 && myOrder[2]==0 && myOrder[3]==3) ||
    (myOrder[1]==3 && myOrder[2]==0 && myOrder[3]==3) ||
    (myOrder[1]==2 && myOrder[2]==1 && myOrder[3]==2) ||
    (myOrder[1]==5 && myOrder[2]==0 && myOrder[3]==4)||
    (myOrder[1]==5 && myOrder[2]==0 && myOrder[3]==3) ||

    length(myOrder[myOrder>4])>0)
      myOrder=c(2,0,2)
```

```
    }
    else
      myOrder=c(2,0,2)
  }
  else
  {
    print("WARNING: Could not determine the model !!! (P, d, q) = (2.0.2)!!!")
    myOrder=c(2,0,2)
  }

  return (myOrder)
}

# calculate_arima_pred : generate one predicted times #series value for a
# considered period 'per' in a considered day 'day', # using arima model
# and 'ndays'as number of day in the training period

# Output=one predicted times series value
#Input= *timeseries=time series values ( contains both train + test values)
#   *ndays:number of days used for the training period,
# Our case Train 2 weeks = 14 days
#   * day :Considered day  for which we want to get the predictions
#   * n: number of periods in the day / our case P=30 min --> n=48 periods
#   * per: considered period in the day
#   * O: arima order used for the prediction
(calculated using calculaModeloArima)
```

```
calculate_arima_pred=function(timeseries, ndays,n,day,o,per){
  datapred=timeseries[(per+n*(day-1)):((ndays*n)+(per-1)+n*(day-1))]
  r <- tryCatch(round(predict(arima(datapred,order=o ), n.ahead =1)$pred),
  error=function(e) e)
  if (is.numeric(r))
  {if(r<0){r=0}else{r=(r)}}else{r=0}
  return(r)
}

#calculate_arima_pred : generate one predicted times series value  for a
#considered period 'per'
# in a considered day 'day', using arima model and 'ndays' as number of day in
#the training period

#Output=one predicted times series value

#Input= *timeseies=time series values ( contains both train + test values)
#       *ndays:number of days used for the training period/
#Our case Train 2 weeks = 14 days
#       * day :Considred day  for which we want to get the predictions
#       * n: number of periods in the day / our case P=30 min --> n=48 periods


calculate_arima_pred_day=function(timeseries, ndays,n,day){
```

```
  ts_train=timeseries[(1+n*(day-1)):((ndays*n)+n*(day-1))]
  o=calculaModeloArima(ts_train)

  r=sapply(1:n, function(x) calculate_arima_pred(timeseries, ndays,n,day,o,x))
  return(r)
}


calculensemble=function(H,pr,pred1,pred2,pred3){

  rtest1=sapply(1:H, function(y)pred1[(pr-H-1+y)])
  xtest=sapply(1:H, function(y) Demand[(pr-H-1+y)])
  rtest2=sapply(1:H, function(y)pred2[(pr-H-1+y)])
  rtest3=sapply(1:H, function(y)pred3[(pr-H-1+y)])

  ro1=sum(abs(rtest1-xtest)/(rtest1+xtest+1))/H
  ro2=sum(abs(rtest2-xtest)/(rtest2+xtest+1))/H
  ro3=sum(abs(rtest3-xtest)/(rtest3+xtest+1))/H
  gama=(1-ro1)+(1-ro2)+(1-ro3)
  r=round((pred1[pr]*(1-ro1)+pred2[pr]*(1-ro2)+pred3[pr]*(1-ro3))/gama)
  return(r)

}

#Demand:Test set : last 50 week =50*7*48=16,800: Vector!!!
```

```
head(df4)

Demand=data[(train_r+1):(t_n_w*7*48)]
#Create tables for the real Demand

s=sapply((1:test_w), function(x) Demand[((x-1)*336+1):(x*336)])

l=lapply((1:test_w), function(x) matrix(s[,x],ncol = 48, byrow = TRUE))

#list containing demand tables for 50 week of the test period
demand_list=lapply((1:test_w), function(x) demandtable(x,l))

#pastdemand:Train set :  98 week =98*7*48=32928: Vector!!!

pastdemand=data[1:train_r]
#Create tables for the past Demand
s1=sapply((1:train_w), function(x) pastdemand[((x-1)*336+1):(x*336)])

l1=lapply((1:train_w), function(x) matrix(s1[,x],ncol = 48, byrow = TRUE))

demand_list1=lapply((1:train_w), function(x) demandtable(x,l1))

n=48
n1=98
n2=50
```

```
pred_p=poisson_pred(n,n1,n2,demand_list1, demand_list)

pred_wp=wpoisson_pred(n,n1,n2,num,we,demand_list1, demand_list)

ts=data[(train_r):(t_n_w*7*48)]

pred_a=unlist(lapply((1:train_w), function(x)calculate_arima_pred_day
(ts, 14,48,x)))

pred_e=NULL
if (is.null(Demand)==TRUE){
  pred_e[1:8]=0
} else{
  pred_e[1:8]=Demand[1:8]
}

pred_e[9:2688]=sapply(9:2688,function(x) calculensemble(8,x,
pred_p,pred_wp,pred_a))
l_res=list(pred_p,pred_wp,pred_a,pred_e)
sMAPE=data.frame(sapply(1:4, function(x) calsmp(2688,l_res[[x]],Demand)))

rownames(sMAPE)=c("Poisson","W.Poisson","Arima","ENS")
colnames(sMAPE)=c("sMAPE")
```

```
library(knitr)
cat('#', 'sMAPE Table', "\n")
cat('\r\n\r\n')
kable(round(sMAPE, digits = 4)*100)
```

(is needed for hyperlinks)

Bibliography

Chang, Han-wen, Yu-chin Tai, Jane Yung-jen Hsu. 2010. Context-aware taxi demand hotspots prediction. *Int. J. Bus. Intell. Data Min.* **5**(1) 3–18. doi:10.1504/IJBIDM.2010.030296. URL `http://dx.doi.org/10.1504/IJBIDM.2010.030296`.

Cramer, Judd, Alan Krueger. 2016. Disruptive change in the taxi business: The case of uber †. *American Economic Review* **106** 177–182. doi:10.1257/aer.p20161002.

Deng, Z., M. Ji. 2011. Spatiotemporal structure of taxi services in shanghai: Using exploratory spatial data analysis 1–5doi:10.1109/GeoInformatics.2011.5981129.

Hall, Jonathan, Cory Kendrick, Chris Nosko. 2015. The effects of uber's surge pricing: A case study. *The University of Chicago Booth School of Business* .

Ihler, Alexander T., Jon Hutchins, Padhraic Smyth. 2006. Adaptive event detection with time-varying poisson processes .

Jason. 2018. A gentle introduction to sarima for time series forecasting in python. `https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/`. (Accessed on 09/30/2019).

Korolko, Nikita, Dawn Woodard, Chiwei Yan, Helin Zhu. 2018. Dynamic pricing and matching in ride-hailing platforms.

Li, Xiaolong, Gang Pan, Zhaohui Wu, Guande Qi, Shijian Li, Daqing Zhang, Wangsheng Zhang, Zonghui Wang. 2012. Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science* **6**(1) 111–121.

MarketWatch. 2017. Ride-hailing industry expected to grow eightfold to $285 billion by 2030 - marketwatch. `https://www.marketwatch.com/story/ride-hailing-industry-expected-to-grow-eightfold-to-285-billion-by-2030`. (Accessed on 10/10/2019).

Moreira-Matias, L., J. Gama, M. Ferreira, L. Damas. 2012. A predictive model for the passenger demand on a taxi network 1014–1019doi:10.1109/ITSC.2012.6338680.

Schneider. 2019. Nyc taxi ridehailing stats dashboard. https://toddwschneider.com/dashboards/nyc-taxi-ridehailing-uber-lyft-data/.

Tang, Haochen, Michael Kerber, Qi-Xing Huang, Leonidas J. Guibas. 2013. Locating lucrative passengers for taxicab drivers .

UberSF. 2018. Uber, lyft main reason for increased traffic congestion in sf, study finds | hoodline. https://hoodline.com/2018/10/uber-lyft-main-reason-for-increased-traffic-congestion-in-sf-study-finds. (Accessed on 11/07/2019).

W.P. Jing, W. Wei, L.K. Hu1. 2015. The recommendation algorithm for taxi drivers based on hadoop and historical trajectory of taxis .

Yu, Bin, Xiaolin Song, Feng Guan, Zhiming Yang, Baozhen Yao. 2016. k-nearest neighbor model for multiple-time-step prediction of short-term traffic condition. *Journal of Transportation Engineering* **142**.

Yuan, Jing, Yu Zheng, Xing Xie, Guangzhong Sun. 2013. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Trans. on Knowl. and Data Eng.* **25**(1) 220–232. doi:10.1109/TKDE.2011.200. URL http://dx.doi.org/10.1109/TKDE.2011.200.

Zhao, Patrick Xuechun, Kun Qin, Qin Zhou, C. Karen Liu, Yi Xin Chen. 2015. Detecting hotspots from taxi trajectory data using spatial cluster analysis .