Predicting ETA Using R

Buan 6356: Final Group Project

Abhishek Dhekane, Abhishek Pandey,
Anupriya Vats, Dilip Merala, Patrick Chou

Table of Contents

Objective ……..…….3

Background …..…….……...3

Motivation …….……..4

Data Exploration ……...5

1. Interpretation of Variables...……..……....6

2. Understanding the Dataset……..……..6

3.Distance Calculation ……..……...…..9

Using OSR Machine ……...……..9

Subsetting the Data ……..10

Prediction ……...12

Using Random Forest Model ……..12

Evaluating the Model ……...13

Testing the Model ……13

Improvements ……..……....15

Conclusion ……....15

Citations ……16

R Code ……...16

OBJECTIVE :

We want to predict the Estimated Time of Arrival (ETA) of Uber cabs in the city of London. Using several key geolocators and calculating the distances between them, we can accurately determine the ETA along the most optimal path. This will help improve efficiency, profitability, and satisfaction for the Uber company, its drivers, and the customers.

BACKGROUND:

Uber has revolutionized the transportation services industry. Since this peer-to-peer ridesharing service began back in March 2009, it has slowly integrated itself into our everyday culture and has also emerged as a major job creator, delivery service, and leader in public transportation. Uber continues to be a frontrunner of the industry by improving on their productivity, usage, and customer relations.

Figure 1. Growth of Uber over its lifetime

	Uber	Lyft	Grab	DiDi
Area of operation	600 cities in 65 countries worldwide	300 US cities, 2 Canadian	Southeast Asia	400 Chinese cities, Brazil, Japan, Mexico, Australia, Hong Kong, Taiwan
Launched	March 2009	June 2012	June 2012	June 2012
Headquarters	San Francisco, US	San Francisco, US	Singapore	Beijing, China
Users	75 million	23 million	36 million	550 million
Drivers	3 million	1.4 million	2.6 million (all time)	21 million
Rides per day	15 million	1 million	4 million	30 million
Total trips	5 billion	500 million	2 billion	7.4 billion in 2017
Revenue	$7.5 billion (2017)	Over $1 billion (2017)	$1 billion (2018 forecast)	$25-27 billion gross; net estimated at 16% of this
Valuation	$72 billion	$15 billion	$11 billion	$56 billion

Figure 2. Comparison of Uber to other companies in the industry

MOTIVATION:

One of the most important features of the Uber app is the estimated time of arrival, which helps us better plan our daily activities. Knowing the right ETA can help us get to meetings on time, fit small but important activities in our busy schedule, and plan around idle time.

As a tech company, Uber refers to this question of predicting ETA as a billion-dollar question. Travel time, or — in their lingo — ETA (estimated time of arrival), is one of the key performance indicators for their business. More often than not, people cancel rides when they see that the ETA is too long. By figuring out the places of maximum ETA, we can make recommendations for improvement which will lead to an increase in the number of trips and help business.

We are curious to build our own model to predict ETA using data for Uber drives in the London area so that we could make recommendations for improving business.

Our prediction and recommendation is for Uber but our model can be applied to any service which is trying to incorporate location intelligence into their service. Retail and wholesale businesses are trying to evaluate catchment areas in their region, real estate companies want to assess locations with their accessibilities, and logistics and cargo companies want to know travel times so they can more efficiently transport goods.

Our mobility assessment needs to be able to create highly accurate travel time predictions with monthly, daily, and even hourly precision for a city of interest. We are using a machine learning approach, so we need a large dataset.

There are numerous supply chain cases where evaluating Google Maps ETA would not be a good fit. Delivery time relies on more than just the length of drop off leg (pickup location to drop location); deliveries involving waiting time needs a calculation of supplementary time for a precise prediction. For instance, food delivery services like Uber Eats also need to predict food preparation time when displaying total delivery time to the customers.

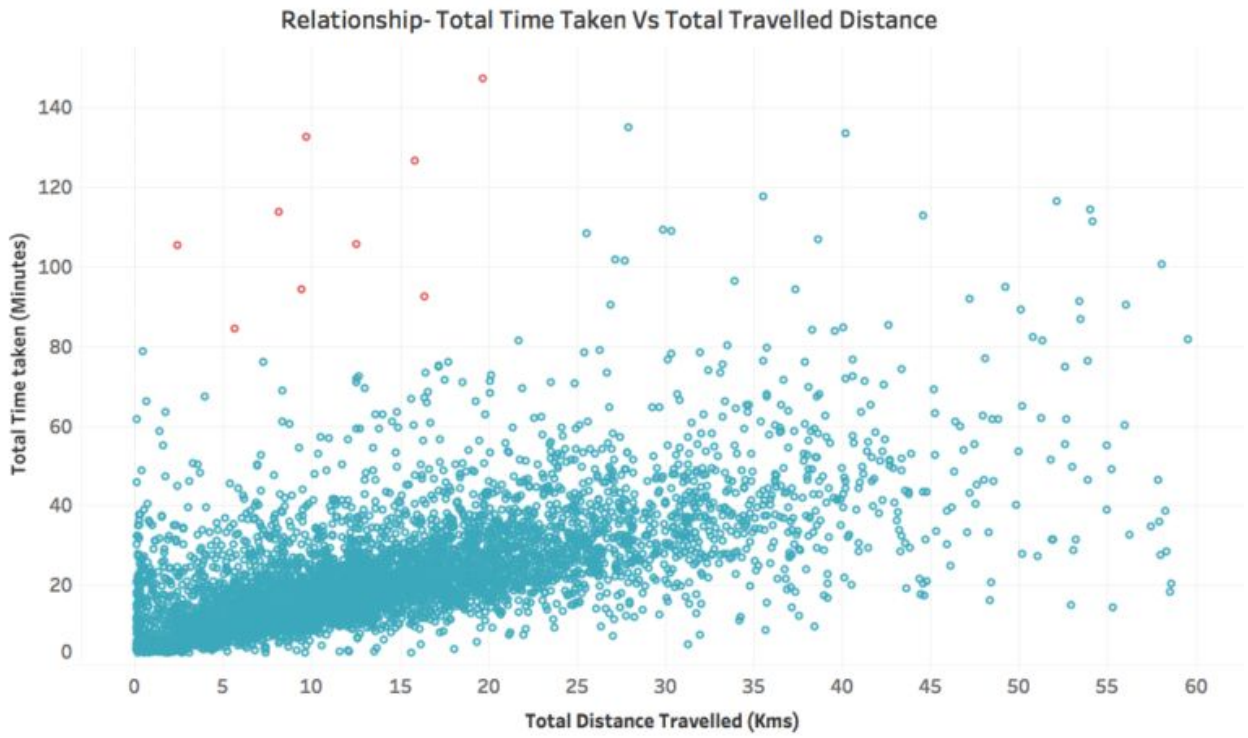

Figure 3. Graph of total time taken vs total traveled distance for UberEats observations

The figure above demonstrates a linear relationship between distance covered and total time taken for the task. It is also clear that there are few rides that took longer to deliver within few kilometers.

These noise in the data need to be removed and other outliers which are incorrectly measured/entered are dropped to improve prediction. Engineering new features influence predictive models by understanding underlying problems. All the above factors inspired us to use Uber's strategic data and build on our own model which can predict time.

DATA EXPLORATION:

We found the following issues with the Uber movement dataset:

1. It did not cover all source and destination pairs for each time interval. There were gaps in it since sometimes there are not enough trips on a given route for them to aggregate and add them to their CSV.
2. It did not provide data aggregated for a specific date-time range in a downloadable format. This means you have an average travel time from origin region to destination region for all Mondays or for 1 pm averaged for 3 months.
3. It was aggregated for districts. It did not have the location (longitude/latitude) of trip start and end points.

Here's our reason behind choosing London as our source city:

Unlike Manhattan or Seattle, London is a big metropolitan area which we thought would be enough to study the city level dynamics unlike the NYC Kaggle dataset which mainly contains Manhattan.

We used the following steps in the data exploration process:

1. Download and explore the weekly aggregated dataset
2. Integrate the weekly aggregated dataset with JSON dataset
3. Apply various models and choose the best from it
4. Visualize the prediction errors on the map. Compare it with the findings in data exploration
5. Compare some travel time results between google maps and the model

Interpretation of final variables used in our dataset:

Origin_lng: Longitude of the origin point
Origin_lat: Latitude of the origin point
Dest_lng: Longitude of the destination point
Dest_lat: Latitude of the destination point
Dow_1: Day of week for which the data is available
Dist: Distance between any origin and destination point pair
Time_1: Mean time taken to travel from an origin point to a destination point using car

Understanding the dataset:

We downloaded the csv file for weekly aggregate dataset from Uber Movement website for London.

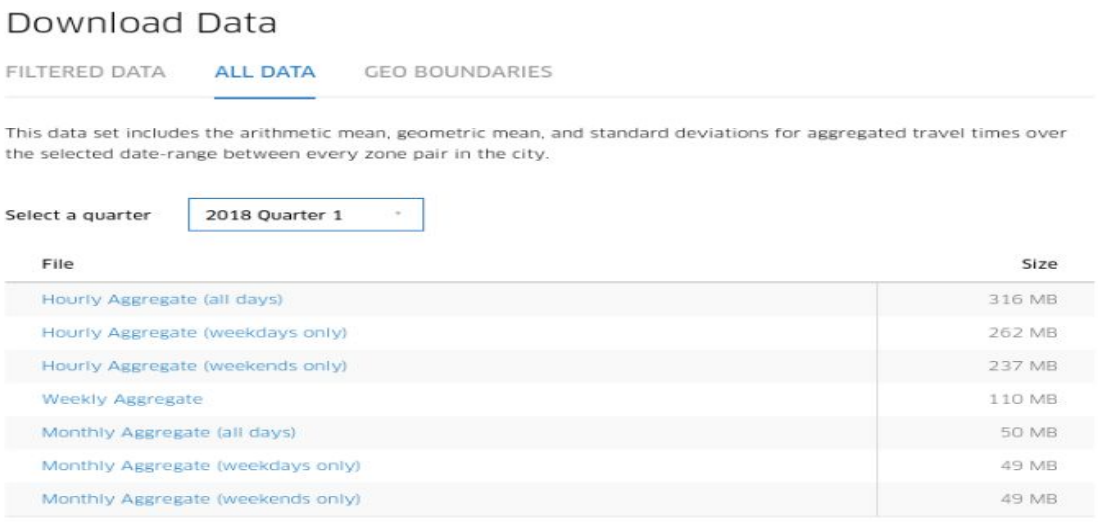

Figure 4.1 Weekly Uber Movement Data

We also needed the geographical boundaries file to set regional origin & destination coordinates.

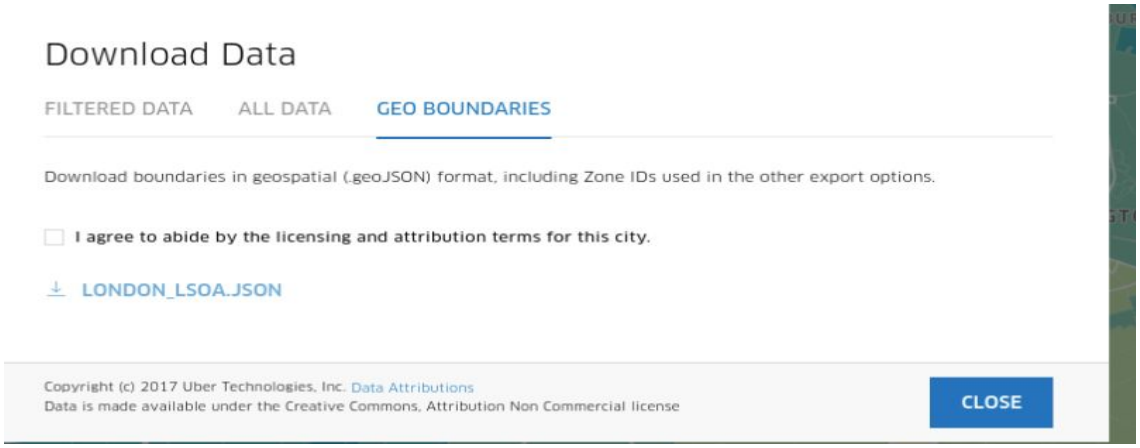

Figure 4.2 JSON File Dataset

Summary of weekly aggregated file:

We have close to 3 million records with 7 variables in our dataset. We can find the mean travel time for each day of week (dow) coded as 1 to 7.

```
str(my_london_mv_wa)

'data.frame':   2885292 obs. of  7 variables:
 $ sourceid                                : int  705 137 131 702
 $ dstid                                   : int  167 201 261 197
131 232 157 451 302 137 ...
 $ dow                                     : int  7 7 7 7 7 1 7 7
 $ mean_travel_time                        : num  1699 1628 3157
 $ standard_deviation_travel_time          : num  600 541 688 206
 $ geometric_mean_travel_time              : num  1621 1556 3088
 $ geometric_standard_deviation_travel_time: num  1.34 1.34 1.23
```

Figure 5. Structure of our Data Frame

We need to map “sourceid” and “dstid”s to regions as we have these IDs as code numbers which need to be changed to origin and destination latitude/longitude to make actual predictions. We read the geoJSON file. It has the definition of 983 regions in London. For each of them, there was a bounding polygon that defined the region. The polygon is created by a list of road segments that define a boundary. Each segment has a start and an end point defined by longitude and latitude.

```r
# Loading objects FROM JSON File
my_london_regions <- jsonlite::fromJSON("london_GB.json")

# Check your region list
head(my_london_regions$features$properties)

# Polygon coordinates for each region
str(my_london_regions$features$geometry$coordinates)

#Calculating centroid of 1 region
my_london_polygons=my_london_regions$features$geometry$coordinates
```

Figure 6. Code for checking polygon coordinates and measuring centroids

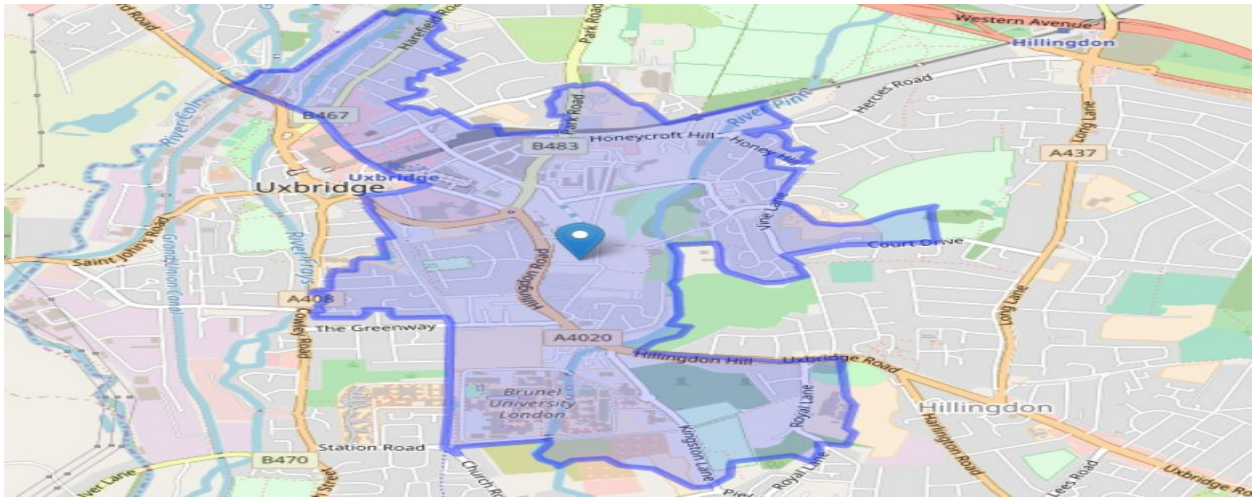

Figure 7. Centroid of 1 region as shown on a map

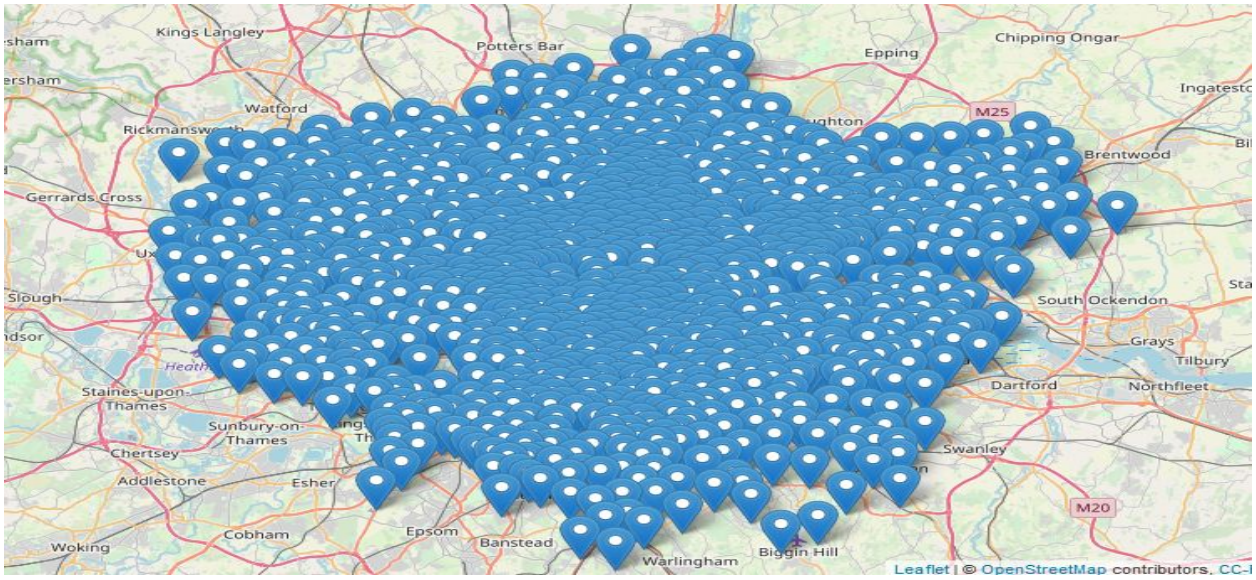

Figure 8. Plot of all 983 regions on the map:

Distance Calculation:

One thing that we are supposed to calculate for our modelling is the distance between any two origin and destination pairs. We cannot rely on Manhattan distance, Euclidean distance, or Crow - Fly distance etc. It has to be the real distance that is travelled by the car, so we are required to use a routing server that can calculate it for us. We use Open Sourcing Routing Machine (OSRM) for the same purpose.

To calculate the distance for the 2.8 million entries, we set up an OSRM server on a laptop and run the data for 72 hours on our machine. Finally, the distances between pick-up and drop-off coordinates are used for the prediction process.

USING THE OSR MACHINE:

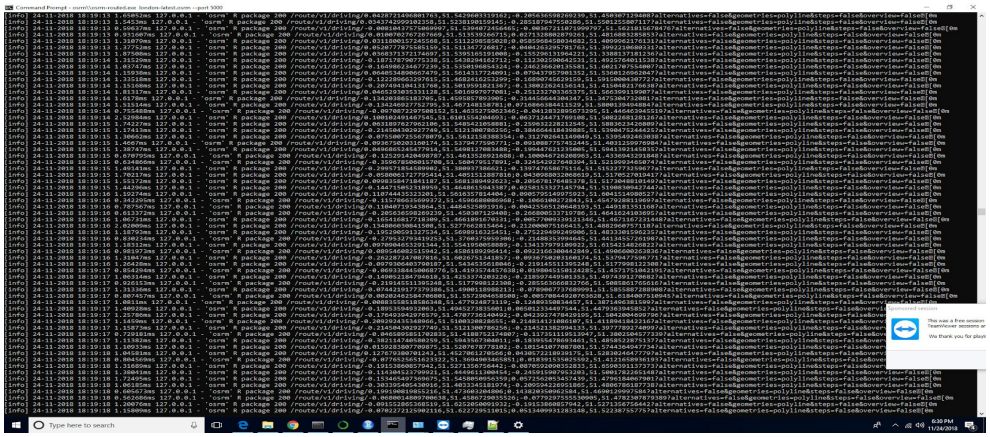

Figure 9. Output of OSR Machine

Steps followed for setting up OSR Machine for windows OS:

1. Create a directory structure, e.g. C:\osrm-data
2. Download github repo of Tom logan using this link. Unzip the folder and copy in the osrm directory created above
3. Download the street network for London using this . Download the .osm.pbf file for Greater London
4. The following command line code needs to be entered in command prompt. It will take 30 minutes to setup before OSRM can be used to generate distance for location pairs.

Step 1 - /osrm/osrm-extract.exe osrm/london-latest.osm.pbf -p /osrm/profiles/foot.luab
Step 2 - /osrm/osrm-contract.exe /london-latest.osrm
Step 3 - /osrm//osrm-routed.exe /london-latest.osrm --port 5000
The above process is used to set up the OSRM server on our computer, to be later tested using the following command:

http://localhost:5000/route/v1/walking/-75.556521,39.746364;-75.545551,39.747228?overview=false

SUBSETTING THE DATA:

While using training models like random forest and LDA we came across problems where if we split our dataset (2.8 million rows) into 70:30 or 80:20 split ratio, our model would not work as our model was exceeding the maximum allocated memory requirements of R.

So we decrease the number of trees in random forest which are explicitly set up to be as 500 to 100, 120 & 50 so that our model uses less memory and give results. Even after decreasing the number of trees, the model took more time than usual to give any results. After trying different

combinations of tree length and train and test set size, we went back and chose 500 random origin and 100 random destination regions which gave us a smaller dataset with around 500*100(routes)*7(dow) = 5,00,000 observations.

Figure 10. New subset data with 500 origin regions

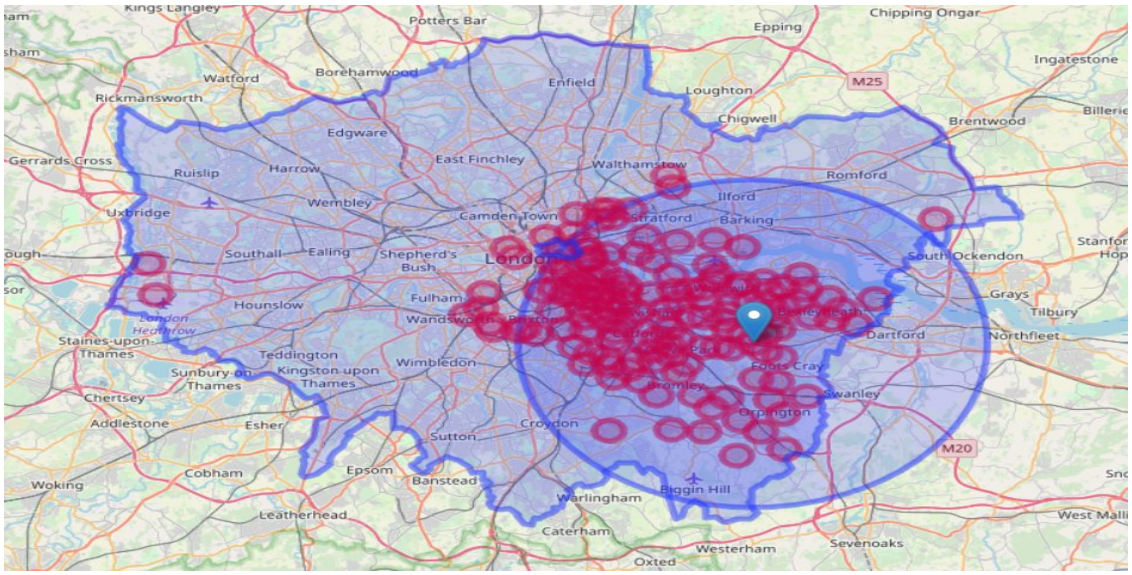

Figure 11. Plot of 100 random destinations

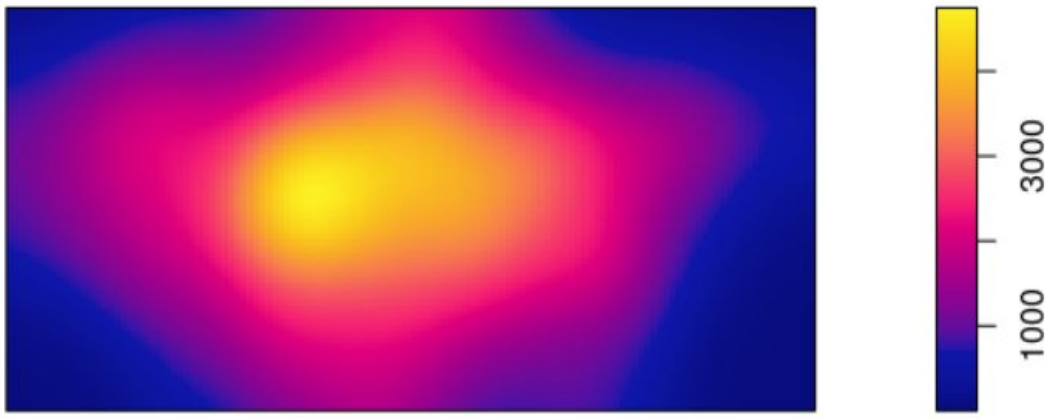

Figure 12. Density plot of origin regions

The density of our origin locations is higher in the center and decreases on the outskirts. Using this we can make a wild guess before modeling and can say that the prediction error will be less in the center since there are many more origin locations (regions) and it will be more on the outskirts.

PREDICTION:

We ran various techniques learned in the BUAN 6356 class on the available data. Eventually we learned and observed the analysis of all the techniques which we used and compared it with each other to get the best fit for the project which came out be as the Random Forest Model.

Initially, we tried running logistics regression on the dataset. For logistics regression, we converted the time values in binary values. For that, we calculated the average time taken by the journeys between two locations from the data and put a constraint of checking the time value of each journey greater than or less than the mean time.

The output gave us the relation between the origin and destination coordinates and time with respect to the mean time taken. The limitation of this model is that it tells us if the time of a particular journey is more or less than the mean time taken according to the dataset. It does not give a definite value of the time taken by a particular journey. Thus we decided to move towards linear regression. After we ran linear regression, we got a very low R-squared value with an RMSE of 654 seconds. We realized that our model is not linear since time varies for different regions depending on origin and source ID combinations. We also did a KNN prediction analysis but came up with an RMSE value of 426 seconds. The RMSE of all the above models, when

compared to the Random Forest RMSE of ~118 seconds, fell short of fitting the criteria of being used in our final model.

Final Prediction using Random Forest Model:

The Random Forest Model turned out to be our best solution. It's an out-of-the-box algorithm which requires minimum feature engineering. We created a training set for the 70% (around 350K rows) of the data set:

```
# Applying random forest to train model
model_Fit <- randomForest(formula = train$time_1 ~ ., data = train, ntree = 120)
```

Result:

Type of random forest : Regressor
No of trees: 120
Mean squared error: 14028.12
Root mean squared error: 118.44

Applying the above model took us 3 hours to generate with training error of 2.12% and test error of 3.6%.

EVALUATING THE MODEL:

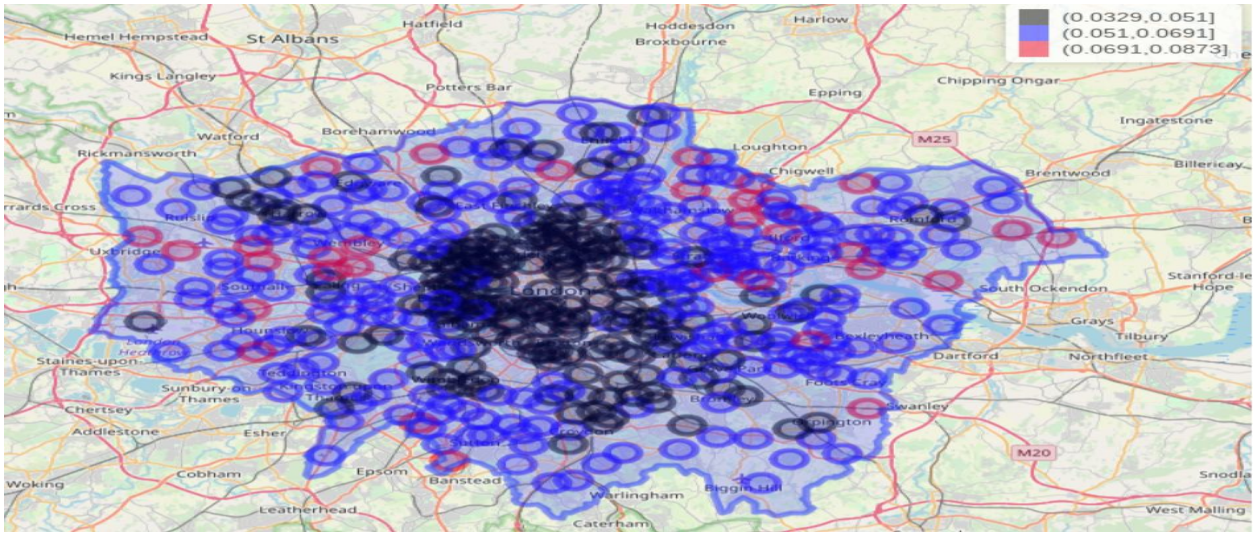

Figure 13. Errors visualized in the test data

As suggested earlier, we can see that there are more errors on the outskirts than in the middle. The black dot shows areas with less errors and red shows the areas with maximum errors.

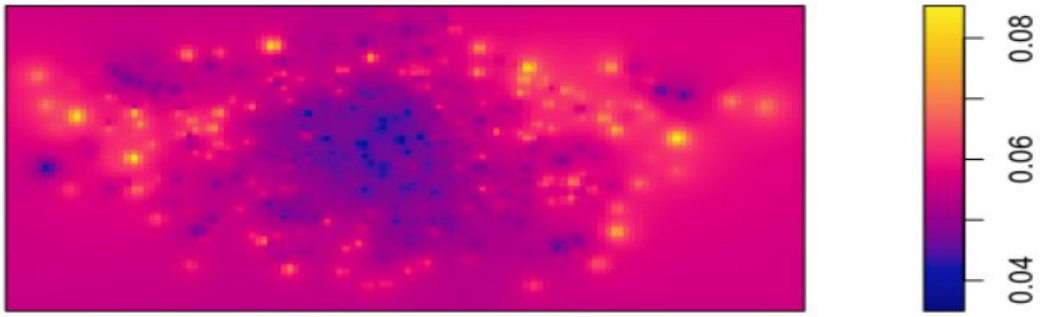

Figure 14. Errors plotted in a density graph

The above density plot also predicts the same that the error in center is less and on outskirts it is more.

Testing the Model:

We then calculated the distance between the very same points with OSRM and passed the required parameters to our model to predict the travel time.

```
42
43  #Testing our model
44  library(osrm)
45  lon_o<-0.054089 ; lat_o<-51.591831
46  lon_d<-0.114256 ; lat_d<-51.553765
47  # calculate distance
48  my_distance<-osrmRoute(src = data.frame(id=1,lon= lon_o , lat=lat_o) ,
49                         dst= data.frame(id=2,lon=lon_d  , lat=lat_d), overview = FALSE)[2]
50  # get route
51  my_route<-osrmRoute(src = data.frame(id=1,lon= lon_o , lat=lat_o) ,
52                      dst= data.frame(id=2,lon=lon_d  , lat=lat_d))
53
54  # calculate travel time with our model for tuesday
55  travel_time<-predict(model_Fit, data.frame(dow_1=2, origin_lng= lon_o,origin_lat=lat_o,
56                                             dest_lng=lon_d ,dest_lat=lat_d,distance=my_distance) )
57
```

Figure 15.1 Code for calculating distance and travel time

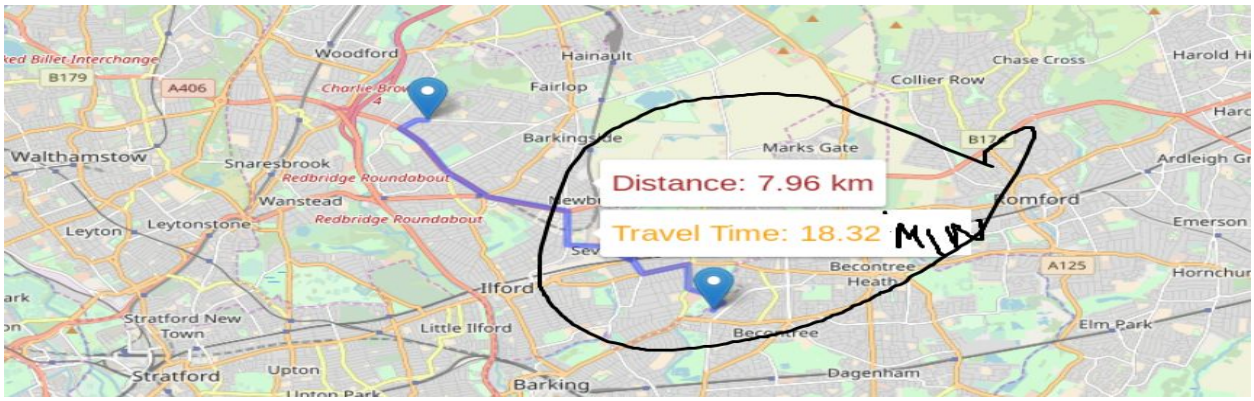

Figure 15.2 Calculated distance and travel time plotted on map

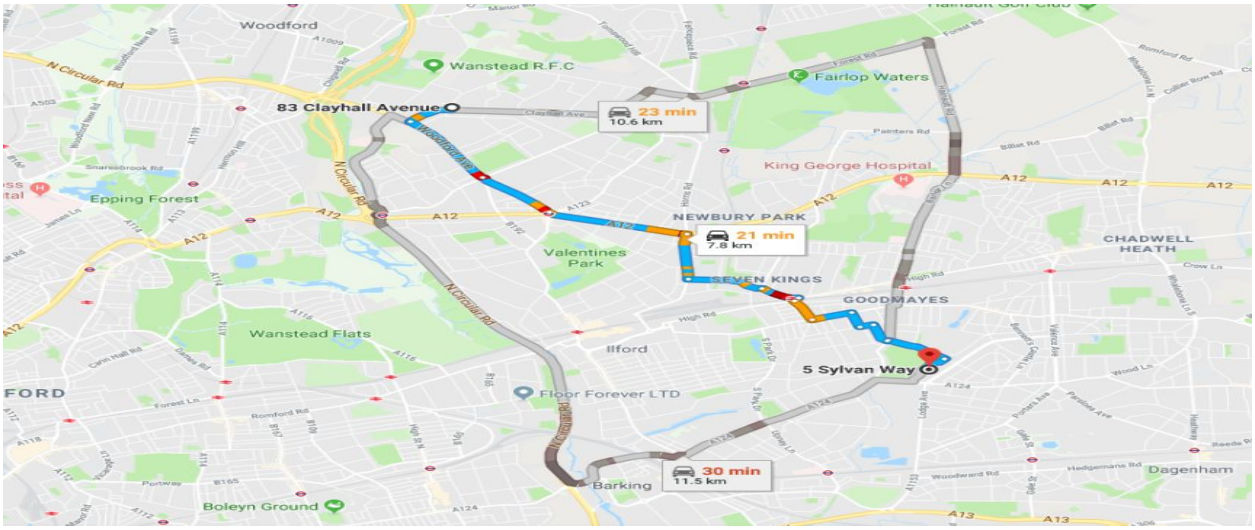

Figure 15.3 Actual distance and travel time on Google Maps

The final results show that our model prediction was close to the actual ETA predicted by google maps with an error rate of around 12.8%. The error rate could rise more for any other random sample of our data. Some of the problems with the model are:

1. It is not aggregated on an hourly level but daily level thus our prediction missed the peak hours time limitation.
2. The error rate has not taken into consideration the seasonality. We took 2018 quarter 1 data used for prediction but calculated the model in other quarter thus we missed the seasonality constraint in our model

Pros of using Random Forest model:

1. It does not require feature engineering i.e. scaling normalization
2. It reduces variance when compared to regular trees or other models
3. Can also be used to extract variable importance

Cons of using Random Forest model:

1. Slows down when the tree length in increased much
2. The results are not that easy to interpret thus we used RMSE to compare different models. Ex: In regression models we know the weightage of each factor but it is not possible to interpret the same in random forest
3. Hyperparameters need good tuning for high accuracy
4. Consumes a lot of memory thus forcing user to subset the dataset

IMPROVEMENTS:

1. We can analyze hourly level data for better and precise predictions.
2. Instead of subsetting the regions randomly, we need to follow some strategic approach.
3. We need to include more outskirts areas for results with less error rate.

4. We can build different models for centers and outskirts. e.g. - More tree depth in random forest on outskirts.
5. Advanced algorithm like XG boosting, neural networks should be used for model computation.
6. We should capture seasonality by working on data at quarter level.

CONCLUSION:

- Spatial analysis is required since we have spatiotemporal data.
- Interpolation is a powerful transformation tool to explore and use such data.
- Selection of origin and destination regions resembles an optimization problem. We are trying to capture the most variability in travel time prediction while holding the origin and destination numbers at the minimum.
- Random forest is a good prediction algorithm but it slows down the overall process of predicting anything with large number of trees.
- It is always a good idea to use different models for different situations. For instance, we should have used different models for center and outskirts.
- Seasonality should always be taken into consideration for more precise results.

CITATION:

Web articles:
Williams, T. (2018). *How to setup an OSRM server*. [online] Reckoning Risk.
https://reckoningrisk.com/coding/2017/OSRM-server/

Iqbal, M. (2018). *Uber Revenue and Usage Statistics (2018) - Business of Apps*. [online] Business of Apps.
http://www.businessofapps.com/data/uber-statistics/#1

Das, A. (2018). *Predicting Arrival time (ETA): Improving Customer's estimation*. [online] JungleWorks.
https://jungleworks.com/predicting-accurate-arrival-time/

Uc-r.github.io. (2018). *Random Forests · UC Business Analytics R Programming Guide*. [online]
https://uc-r.github.io/random_forests

Company Website:
Uber Movement. 2018

https://movement.uber.com/explore/london/travel-times/query?lang=en-US&lat.=51.51262&lng.=-0.130438&z.=12&si=573&ti=&ag=lsoa&dt[tpb]=ALL_DAY&dt[wd;]=1,2,3,4,5,6,7&dt[dr][sd]=2018-01-01&dt[dr][ed]=2018-01-31&cd=

Osrm setup:
https://github.com/tommlogan/city_access

APPENDIX - R CODE:

```
#Loading libraries
library(jsonlite)
library(geosphere)
library(tidyverse)
library(leaflet)
library(ggplot)
library(ggmap)
library(spatstat)
library(osrm)
library(sqldf)
library(randomForest)
library(caret)
library(caTools)

#Setting up working directory
setwd("C:/Users/abhip/OneDrive/Desktop")

#Reading csv file
London_csv <- read.csv("London_UM.csv")

# Checking the structure of data
str(London_csv)

# Loading objects FROM JSON File
my_london_regions <- jsonlite::fromJSON("london_GB.json")

# Check your region list
head(my_london_regions$features$properties)

# Polygon coordinates for each region
str(my_london_regions$features$geometry$coordinates)

#Calculating centroid of 1 region
my_london_polygons=my_london_regions$features$geometry$coordinates
```

```
#Creating an empty dataframe
dat <- data.frame()

#Looping to store centroids of all the observations
for (i in 1: length(my_london_polygons)){
  my_temp_poly<-my_london_polygons[[i]]
  poly_len <- length(my_temp_poly)/2
  poly_df <- data.frame (lng = my_temp_poly[1,1,1:poly_len,1], lat =
my_temp_poly[1,1,1:poly_len,2])
  my_poly_matrix <- data.matrix(poly_df)
  temp_centroid <- data.frame()
  temp_centroid <- centroid(my_poly_matrix)
  temp_centroid[[3]] <- i
  dat <- rbind(dat, temp_centroid)
}

#Changing column names
colnames(dat) <- c("lng", "lat", "id")

#Changing the order of columns
dat <- dat[c("id", "lng", "lat")]

#Plotting regions on graph
leaflet(dat) %>%
  addTiles() %>%
  addMarkers() %>%
  addPolygons(lng= poly_df$lng, lat=poly_df$lat)

#Merging csv and dat on the basis of source id to add origin lat and long
final_df_file <-
sqldf("SELECT dow as dow_1, dstid as dstid_1, mean_travel_time as time_1, lng as origin_lng,
lat as origin_lat
  FROM London_csv d1 JOIN dat d2
  ON d1.sourceid = d2.id")

#Merging csv and dat on the basis of destination id to add origin lat and long
final_df_file_07 <-
  sqldf("SELECT dow_1, time_1, origin_lng, origin_lat, lng as dest_lng, lat as dest_lat
  FROM final_df_file d5 JOIN dat d6
  ON d5.dstid_1 = d6.id")

#Triggering the osrm server to extract the distance between points
options(osrm.server = "http://127.0.0.1:5000/")
```

```
emp_dat <- data.frame()
for (j in 1: length(final_df_file_07$dow_1)){
print(j)
my_route<- osrmRoute(src=final_df_file_07[j,c(1,3:4)], dst=final_df_file_07[j,c(1,5,6)], overview
= FALSE)
emp_dat <- rbind(emp_dat, my_route)
}

colnames(emp_dat) <- c("duration", "dist")

#Making final dataframe for modelling
ml_df <- cbind(final_df_file_07, emp_dat)

#Temporary model evaluation:
model_01_df <- ml_df[-c(7)]
#Changing sequence of columns in our temporary dataset to be trained
model_01_df <- model_01_df[c("origin_lng", "origin_lat", "dest_lng", "dest_lat", "dow_1", "dist" ,
"time_1" )]


#Splitting dataset for training & modelling

sample = sample.split(model_01_df$dow_1, SplitRatio = .60)
train = subset(model_01_df, sample == TRUE)
test  = subset(model_01_df, sample == FALSE)


# Applying random forest to train model
model_Fit <- randomForest(formula = train$time_1 ~ ., data = train, ntree = 120)

# Predicting the Test set results
y_pred = predict(model_Fit, newdata = test[-7])

# Calculating Accuracy of model using  RMSE
RMSE <- sqrt(mean((y_pred - test[,7])^2))

#Testing our model
library(osrm)
lon_o<-0.054089 ; lat_o<-51.591831
lon_d<-0.114256 ; lat_d<-51.553765
# calculate distance
my_distance<-osrmRoute(src = data.frame(id=1,lon= lon_o , lat=lat_o) ,
                       dst= data.frame(id=2,lon=lon_d  , lat=lat_d), overview = FALSE)[2]
```

```
# get route
my_route<-osrmRoute(src = data.frame(id=1,lon= lon_o , lat=lat_o) ,
                    dst= data.frame(id=2,lon=lon_d  , lat=lat_d))


# calculate travel time with our model for tuesday
travel_time<-predict(model_Fit, data.frame(dow_1=2, origin_lng= lon_o,origin_lat=lat_o,
                                           dest_lng=lon_d ,dest_lat=lat_d,distance=my_distance) )
```

BUAN 6356 - Abhishek Pandey: Project Individual Report

I have always believed in a quote: **Innovation is not an activity, it's a mindset**. Taking the same mindset we went on to work on our project where we tried predicting Expected time of arrival of an Uber cab in London city. I come from a background where I had a chance to work for one of the most dynamic companies in India, one of them being Olacabs. I was a part of the analytics team where I worked on various projects but I never got a chance to build anything from the scratch that can predict and suggest anything.

Predicting the expected time of arrival has always been one of the most crucial pain points for any company in logistics and transportation industry. Ex - Optimizing routes to decrease time of delivery by 30 seconds on each delivery can help UPS to save $ 10 million in a year. This is just an example, and these companies are always striving to optimize and predict the right ETA which helps them saving big money. The best source of data for our project turned out to be Uber movement data which is officially published by Uber company. Using their data meant that we were indirectly working on the live dataset.

According to a report by the Beneson Strategy Group, 73% of Uber drivers prefer having a job that lets them choose their schedule. Drivers can use this flexibility to maximize their expected revenue during their available times. To this end, Uber provides heatmaps of customer demand that allow drivers to target high-demand regions that provide higher trip probability and therefore higher expected revenues.

However, there is no readily available way to determine whether trips starting at a particular location are worth the time it would take to service the trip. For example, a trip that takes longer due to traffic but has high demand may result in fewer total trips and be less valuable. On the other hand trips with less demand but are consistently much faster may be more valuable. Therefore it is in the driver's best interest to balance the demand of a location and the amount of time trips from that location will take.

Considering the above factors we wanted to find the optimized revenue that a driver can earn according to his time schedule. This project required advanced R skills to do initial cleaning and later build model according to driver requirements. We were required to have a knowledge of integer programming to get done with our project. Also, most of my teammates were not able to clean the data and understand the purpose of this project thus we decided to change our project midway and make it a little easier.

We then went on to predict the ETA of uber cabs, which was in no way an easy task but easier then what we were trying to accomplish earlier. The main problem that I faced in this project was to generate the optimized distance using OSR machine and then later training our model on huge dataset (2.8*7 million). Setting up the OSR machine was a great experience and it will help me with my future projects as well. While training my model in R for such a huge dataset my memory was getting exhausted without subsetting of data. I realized that python is a better alternative when one is trying to use machine learning methods like random forest on big dataset. I also came to know about a few new algorithms like XG boosting, neural networks etc which could work more efficiently on such a huge dataset.

The overall experience of working on this project in a team was good. I got a chance to mentor or guide people in team. I was happy to see that everyone wanted to contribute their bit with whatever knowledge they had. I am someone who is not good at formatting and finalizing the final reports and I would like to thank my team for teaching me how to go about it, specially Dilip and Patrick who are responsible for how our project report is looking like at the moment.

I would be continuing with this project and extending it to do what our initial aim of the project was i.e. take drivers schedule into consideration and suggest the best way to earn optimized revenue accordingly.

Lastly, I would like to thank Prof. Sourav for introducing us with all these machine learning methods. The basic introduction of these topics in class helped me to go and search more about these algorithms at home. I would appreciate if you could introduce more assignments like the third assignment. It was a great experience to work on such an assignment. One last suggestion that I would want to make is datacamp is not that intuitive. If we could get to work on platforms like dataquest which have intuitive projects at the end of each section then it would help students to learn in more efficient way. Thank you for giving us the opportunity to work on this project.