

A web application:

Idle Time Minimization of Uber Driver

A Project submitted to the
Department of Computer Science and Engineering, Jahangirnagar University
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

By

Arafat Rahman Exam Roll: 160025 Registration No: 41590 Session: 2015-2016	Asfakul Ghani Exam Roll: 160041 Registration No: 41607 Session: 2015-2016
--	--

Supervised by
Tahsina Hashem
Assistant Professor
Department of CSE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JAHANGIRNAGAR UNIVERSITY

JANUARY 2021

ABSTRACT



With the immense popularity of ride sharing services, more and more people use services like Uber. It is an online marketplace for riders and drivers. Normally a rider uses his smartphone app to request rides. Ride requests are assigned to a Uber driver, who uses his own vehicle to provide the ride. Low cost, short waiting time, availability as well as the convenience of simplified ride requests and easy payment are considered the main reasons of Uber's popularity among the riders. On the other hand, the flexibility of work schedule, higher compensation rates and independence attract a lot of part time and full time drivers to contract with Uber. Students, professional drivers or people in-between jobs share their ride to earn some extra cash. As an independent worker, a driver would always want to fill the car with passengers to maximize profit. But it is not always easy to find a rider. Especially for a new driver in a new city. We analyzed Uber historic data in New York city and built a web application. Based on the holiday or workday and different time of the day, we suggest a location to the driver which is not very far away from the current location and has the higher probability of getting a ride.

DECLARATION

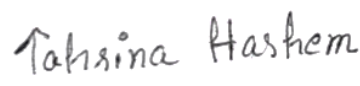
The project work entitled “**Idle Time Minimization of Uber Driver**” has been carried out in the Department of Computer Science and Engineering, Jahangirnagar University is original and conforms to the regulations of this University.

I understand the University’s policy on plagiarism and declare that no part of this project has been copied from other sources or been previously submitted elsewhere for the award of any degree or diploma.

Declared By

 (Arafat Rahman)	 (Asfakul Ghani)
--	---

Counter Signed by


(Supervisor)

ACKNOWLEDGEMENT

We express our heart-felt gratitude to our supervisor, Tahsina Hashem for her constant supervision of this work. Under her supervision we have learned so many new things in the process of completing this project. We are forever grateful to have the opportunity to work with her.

In this regard, we remain grateful to our beloved parents, who always exist as sources of inspiration behind every success of ours we have ever made.

Contents

Abstract	ii
Declaration	iii
Acknowledgement	iv
List of Figures	vii
List of Tables	viii
Chapter 1: Introduction	1
1.1 Introduction	1
1.2 Motivation	2
1.3 Methodology	3
1.3.1 Frontend	3
1.3.2 Backend	3
1.4 Contribution	3
1.5 Outline	4
Chapter 2: Related Works	5
Chapter 3: Data	7
3.1 Data Collection	7
3.2 Data Processing	7
Chapter 4: Frameworks and Tools	9
4.1 Python Libraries	9
4.1.1 Pandas	9
4.1.2 Scikit-learn	9
4.1.3 Matplotlib	10
4.2 Django (Python Framework)	11
4.2.1 Installing and Creating a Project	11
4.2.2 Starting Web Server	12

4.3 Mapquest API-key	13
Chapter 5: System Design and Implementation	15
5.1 K-means Clustering	15
5.1.1 K-means Clustering Method	15
5.1.2 How to Choose K	17
5.1.3 K-means Cluster in Python Library	17
5.1.3.1 Function Declaration	17
5.1.3.2 Function Parameters	17
5.1.3.3 Function Attributes	18
5.2 R-tree	19
5.2.1 Properties of R-tree	19
5.2.2 R-tree in Python Library	21
5.3 Design	21
Chapter 6: Result Analysis	24
6.1 Output	24
6.1.1 Initial Map	24
6.1.2 Path Direction	25
6.1.3 Hour Input	26
6.1.4 Holiday Input	27
6.1.5 Stamen Terrain Basemap	28
6.2 Accuracy of Our System	29
Chapter 7: Conclusion	31
7.1 Conclusion	31
7.2 Future Work	31
References	32

LIST OF FIGURES

1.2.1 Uber pickups	2
3.2.1 Different pickup frequency on different hour of the day and holiday	8
4.2.1.1 Creating a virtual environment	11
4.2.1.2 Starting a django project	12
4.2.2.1 Starting the web server	12
4.2.2.2 Web server in the localhost	13
4.3.1 Map API	13
5.1.1.1 K-means clustering	16
5.2.1.1 R-tree	20
5.3.1 Comparison of different number of clusters	22
6.1.1.1 Cluster center and cluster region (Snapshot of our System)	24
6.1.2.1 Route-Map view of our System	25
6.1.3.1 Route-Map view (Hour = Selected by dropdown menu)	26
6.1.4.1 Route-Map view (Holiday = Selected by radio-button feature)	27
6.1.5.1 Route-Map view (Stamen Terrain Basemap)	28
6.2.1 Accuracy and Time Complexity for different amounts of closest stations(T)	30

LIST OF TABLES

6.2.1 Accuracy and Time by considering different amount of closest station	29
--	----

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Now-a-days a large number of people use ridesharing services like Uber, Lyft. According to Uber, their service is available in 900 cities, across 93 countries. There are 103 million Uber monthly average users who are served by a total of 5 million drivers^[1].

Uber, founded in 2009, is one of the most successful ridesharing companies. It is an online marketplace for riders and drivers. Normally, a rider uses his smartphone Uber app to request a ride. The ride requests are assigned to a specific Uber driver who uses his own vehicle to provide the ride. Low cost, short waiting time, availability as well as the convenience of simplified ride requests and easy payment are the main reasons contributing to Uber's popularity among the riders. On the other hand, the flexibility of work schedule, higher compensation rates and independence are among the main reasons making Uber popular with drivers.

Independence of work attracts a lot of part time and full time drivers to contract with Uber. Students, professional drivers or people in-between jobs share their ride to earn some extra cash. As an independent worker, a driver would always want to fill the car with passengers to maximize profit. But it is not always easy to find a rider. Especially for a new driver in a new city. We analyzed Uber historic data in New York city and built a web application. We suggest a place to the driver to be where he can find a rider.

1.2 MOTIVATION

A rider requests a ride using Uber App. A nearby Uber driver can accept the request and complete the ride. In order to accept the ride request a driver has to be in a certain radius of the rider when the rider requests the ride. If the driver is far away, he won't be able to accept the ride request. And if a driver doesn't get a request, he has to stay idle or roam around randomly.

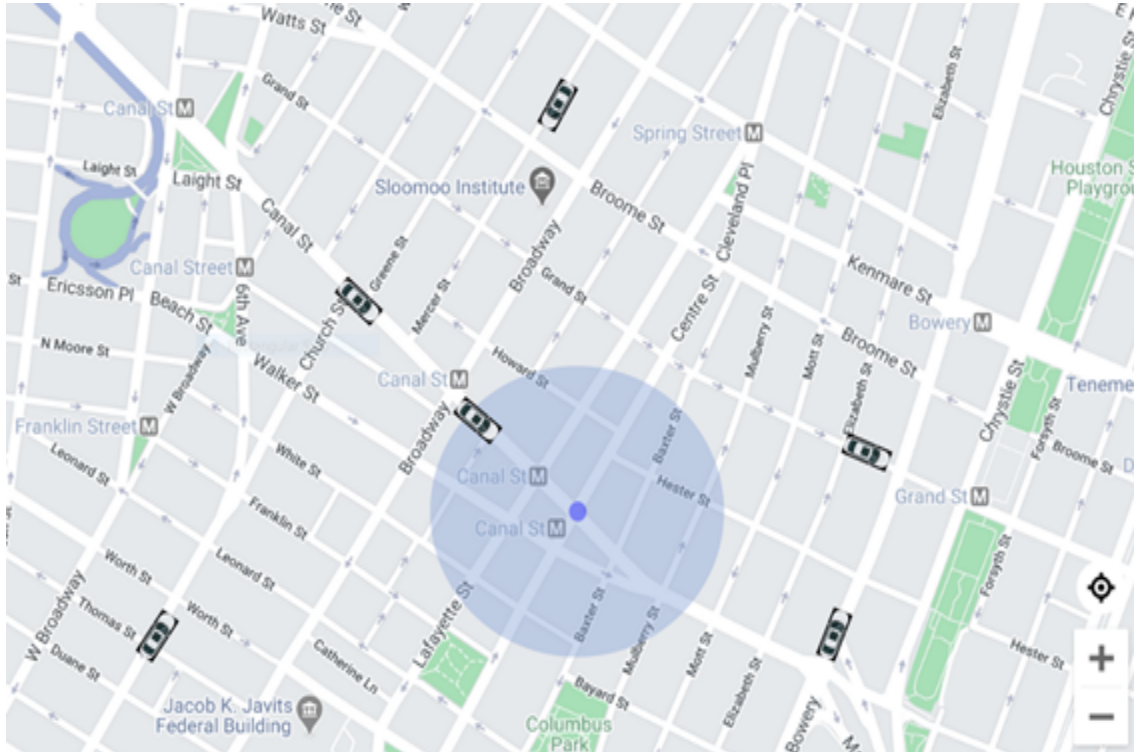


Figure 1.2.1 Uber pickups

In our web application, based on Uber historic data we suggest an optimal location for the driver where:

1. **It is highly likely to have a ride request.**
2. **Not very far away from the current location.**

1.3 METHODOLOGY

1.3.1 FRONTEND

In the frontend we provide an interactive map. A user can click the map and set the clicked location as his current location. We also provide some forms to input other parameters like time of the day and holiday or workday.

In the output we provide the destination location where he should go to find a rider. We also provide a shortest path to reach the destination from current location, time to reach the destination as well as a guide to reach the destination.

1.3.2 BACKEND

In the backend we train the model using Uber historic data. We used a clustering algorithm to train the model faster by reducing the total number of pickup locations. Also we used a spatial data structure to answer the user's query faster.

1.4 CONTRIBUTION

We analyzed the Uber historic data and built a web application. Based on the data we recommend not only an optimal location but also provide a step by step guide on how to reach the destination in the shortest possible path.

We used K-means clustering algorithm to form 'pickup stations' with nearby pickup points. By that we have been able to reduce the pickup points to 10% and allowed us to work with a large set of data.

We used a spatial data structure R-tree^[9]. It is used to quickly find the closest coordinates of a given coordinate. Thus we have been able to process users' queries faster.

1.5 OUTLINE

The remainder of the project report is organized as follows. In Chapter 2, we review the existing related works to this project work. Chapter 3 gives an overview of real dataset collection and processing. In Chapter 4 we illustrate the framework and the tools in detail that we have used. Chapter 5 describes our entire process of designing our web application. In Chapter 6, several snapshots and thorough guidelines of our web application are presented. Chapter 7 concludes the research work with future goals and directions.

CHAPTER 2: RELATED WORKS

1. Uber Optimization: Finding Passengers Faster.^[2]

They recommend a destination for the driver based on the location, time of the day, day of the week and weather. They find all the pickup points in a quarter mile radius around the user's location. For each of those pickup points they calculate the score of the point by the number of other pickup points within a tenth of a mile radius. And the best score pickup point is selected as the destination.

2. Uber/Lyft Maximization: More Money for The Time.^[3]

They transformed the pickup and dropoff coordinates into a heatmap using Leaflet plugin. They also added filters so that the user can manually select the days of the week and times for pick-ups or drop-offs. To give drivers a better idea they added a histogram on the sidebar to show the number of pick-ups and drop-offs on the users filtered criteria.

3. Reinforcement Learning for Optimizing Driving Policies on Cruising Taxis Services.^[4]

To decrease waiting time of a cruising taxi driver, they simulated different decisions of the driver. For that, they built a Reinforcement Learning framework using dynamic programming. They formulated a Markov Decision Process on driver's behavior considering the effect of driver's action in the long run.

4. Optimizing Earnings for On-Demand Ride-Hailing.^[5]

Through a series of dynamic programming, they optimize a strategy for a ride-hailing driver to maximize the expected earning.

5. Spatio-temporal feature fusion for Dynamic taxi route recommendation.^[6]

To decrease waiting time of a passenger and increase profit of drivers they recommend a dynamic taxi route to the drivers. First they use spatio-temporal features to measure the degree of easiness of a vacant taxi to pick up a new passenger. Second, to recommend the route effectively they design an adaptive deep reinforcement learning method to better fuse the extracted spatio-temporal features.

6. A Cost-Effective Recommender System for Taxi Drivers.^[7]

They developed a cost-effective recommender system for taxi drivers. For this, they use a brute-force strategy to find the optimal route for recommendation.

7. Route Recommendations for Idle Taxi Drivers.^[8]

Their main goal is to minimize the distance between the taxi driver and the next anticipated passenger. To anticipate the next passenger, they used a Monte Carlo Tree Search and developed a route recommendation engine called MDM:Minimizing Distance. Simulation shows that the model is robust to anomalous events like concerts, sporting events, etc.

CHAPTER 3: DATA

3.1 DATA COLLECTION

In 2015, by filing a Freedom of Information Law request the news and analytics website FiveThirtyEight obtained historical Uber data in New York City from the NYC Taxi & Limousine Commission. From this data set we used pickups from April, 2014 - September, 2014.^[10]

In this dataset we have GPS coordinates of the pickup location as well as time and date when the pickup happened.

We used another dataset from data.world^[11] to verify if the corresponding pickup date was a holiday or not in New York City.

3.2 DATA PROCESSING

We consider three attributes in our real dataset.

1. **Location** is a pair of float variables indicating the Latitude and Longitude. In our dataset Location is from where Uber picked up the passenger. In the output we also provide a location as the destination.
2. **Hour** is the time of the day when pick up has happened. We divide the day in 24 hours. So the Hour variable can take values between 0-23. We need Hour as a different attribute because depending on the hour of the day pickup frequency can differ.
3. **Holiday** is a boolean variable. True if the day is a holiday and false if not. Also depending on the holiday pickup frequency can differ. During the holiday people tend to stay awake late at night.

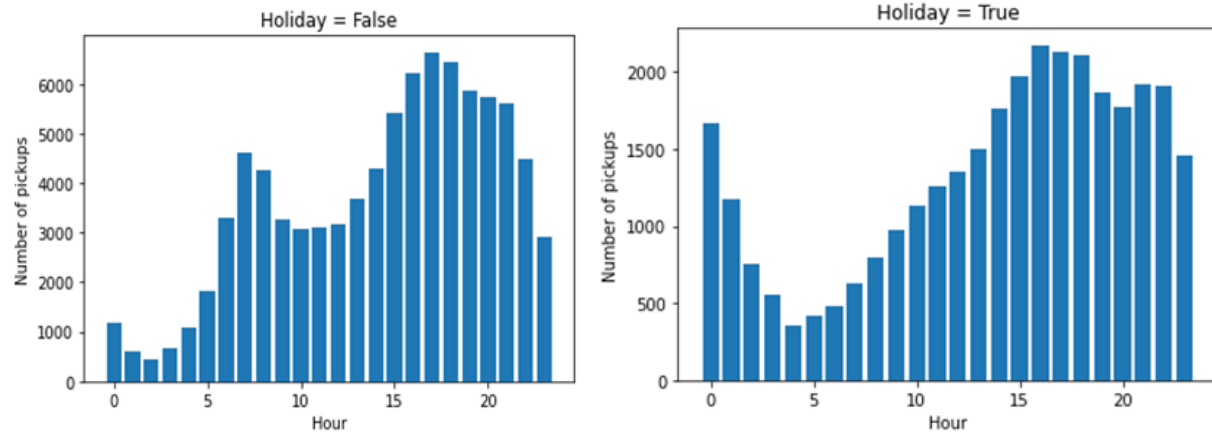


Figure 3.2.1 Different pickup frequency on different hour of the day and holiday.

After cleaning and merging the data, we have a total of 4.5 million Uber pickup points in New York City. We chose a subset (around 3%) of the data randomly for our project.

CHAPTER 4: FRAMEWORKS AND TOOLS

4.1 PYTHON LIBRARIES

4.1.1 PANDAS

Pandas is an open source Python library for data analysis as well as in data science. The package is used for various data manipulation tasks. It includes a variety of techniques which is much easier working on data analysis related problems.

Features of pandas dataframe:

- ☐ Columns can be different types of data.
- ☐ Size can be altered.
- ☐ Rows and Columns are arranged in label based.
- ☐ Different types of arithmetic operations can be done.

Installation using pip: *pip install pandas*

4.1.2 SCIKIT-LEARN

Scikit-learn is the most strong Python library for machine learning. It is used for solving many machine learning and statistical modeling problems including classification, regression, clustering etc. in Python.

Pre-requisites:

- ☐ Python (Accessible for version ≥ 3.5)
- ☐ NumPy (Accessible for version $\geq 1.11.0$)
- ☐ Scipy (Accessible for version $\geq 0.17.0$)
- ☐ Matplotlib (Accessible for version $\geq 1.5.1$)
- ☐ Pandas (Accessible for version $\geq 0.18.0$)

Scikit-learn provides including:

- ❑ Supervised learning algorithms (such as linear regression, support vector machine(SVM), decision tree etc.)
- ❑ Unsupervised learning algorithms (such as clustering etc.)
- ❑ Different kinds of Preprocessing such as Min-Max Normalization
- ❑ Reduce the number of features and create some new features (feature extraction)
- ❑ Select those features which are most relevant (feature selection)

Installation using pip: *pip install -U scikit-learn*

4.1.3 MATPLOTLIB

Matplotlib is a popular Python library which is used for data visualization including histograms, scatterplots etc. It mainly plots the dataset into 2-D visualization format. In January 2018, the latest version (2.2.0) was released.

Histogram plotting: The *matplotlib.pyplot.hist()* function draws a histogram including some computations.

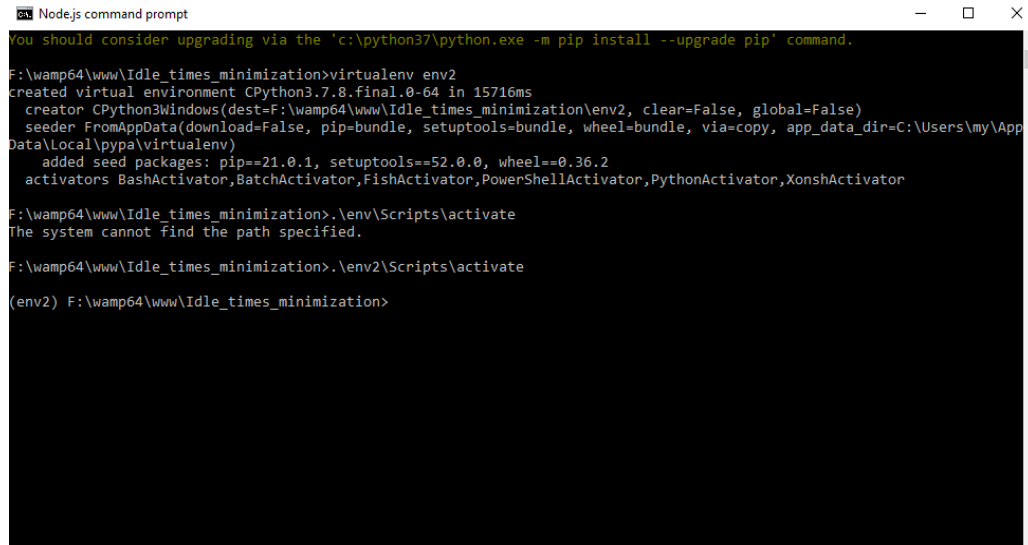
Scatter-plotting: Scatter plots are used to plot data points on horizontal and vertical axis to visualize how data points look like.

4.2 DJANGO (PYTHON FRAMEWORK)

Django is a widely used and high level Python web framework which provides the MVT (model-templates-views) architecture pattern. It is a fully featured server-side framework and provides the full stack services as well. Simplicity, flexibility, reliability and scalability are the main goals of this web framework.

4.2.1 INSTALLING AND CREATING A PROJECT

In order to install django, first we need to create a virtual environment. Because we don't want to disturb our whole system.



```
Node.js command prompt
You should consider upgrading via the 'c:\python37\python.exe -m pip install --upgrade pip' command.

F:\wamp64\www\Idle_times_minimization>virtualenv env2
created virtual environment CPython3.7.8.final.0-64 in 15716ms
  creator CPython3Windows(dest=F:\wamp64\www\Idle_times_minimization\env2, clear=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\my\AppData\Local\pypa\virtualenv)
    added seed packages: pip==21.0.1, setuptools==52.0.0, wheel==0.36.2
  activators BashActivator,BatchActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator

F:\wamp64\www\Idle_times_minimization>.\env\Scripts\activate
The system cannot find the path specified.

F:\wamp64\www\Idle_times_minimization>.\env2\Scripts\activate
(env2) F:\wamp64\www\Idle_times_minimization>
```

Figure 4.2.1.1 Creating a virtual environment

After creating the virtual environment we can install django and start a new project.

```
Nodejs command prompt
activators BashActivator, BatchActivator, FishActivator, PowerShellActivator, PythonActivator, XonshActivator
F:\wamp64\www\Idle_times_minimization>.\env\Scripts\activate
The system cannot find the path specified.
F:\wamp64\www\Idle_times_minimization>.\env2\Scripts\activate
(env2) F:\wamp64\www\Idle_times_minimization>pip install django
Collecting django
  Using cached Django-3.1.7-py3-none-any.whl (7.8 MB)
Collecting asgiref<4,>=3.2.10
  Using cached asgiref-3.3.1-py3-none-any.whl (19 kB)
Collecting pytz
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.1-py3-none-any.whl (42 kB)
Installing collected packages: sqlparse, pytz, asgiref, django
Successfully installed asgiref-3.3.1 django-3.1.7 pytz-2021.1 sqlparse-0.4.1
(env2) F:\wamp64\www\Idle_times_minimization>django-admin startproject backend
(env2) F:\wamp64\www\Idle_times_minimization>cd backend
(env2) F:\wamp64\www\Idle_times_minimization\backend>
```

Figure 4.2.1.2 Starting a django project

4.2.2 STARTING WEB SERVER

Our created project will run in the local web server.

```
Nodejs command prompt - python manage.py runserver
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying sessions.0001_initial... OK
(env2) F:\wamp64\www\Idle_times_minimization\backend>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
February 27, 2021 - 21:25:13
Django version 3.1.7, using settings 'backend.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 4.2.2.1 Starting the web server

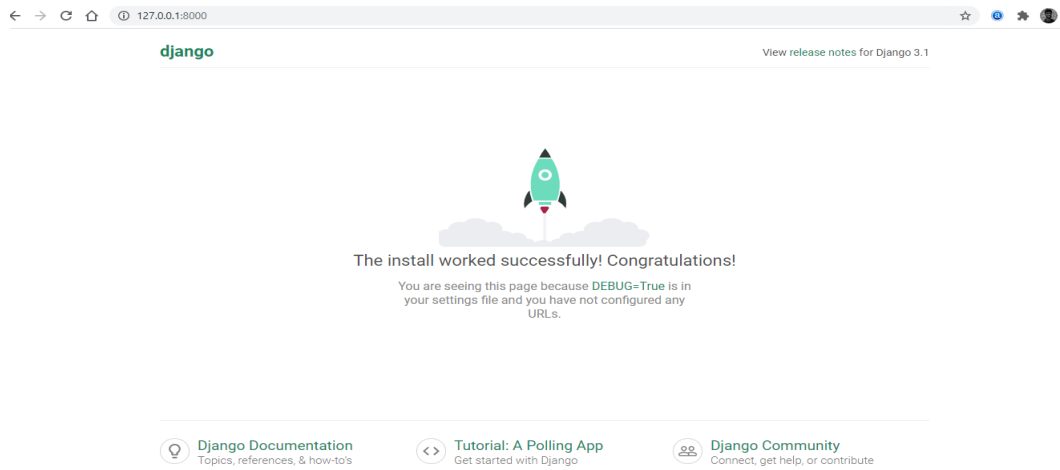


Figure 4.2.2.2 Web server in the localhost

4.3 MAPQUEST API-KEY

We used Mapquest API-key to take input from the user and also output the destination as well as direction to the destination.

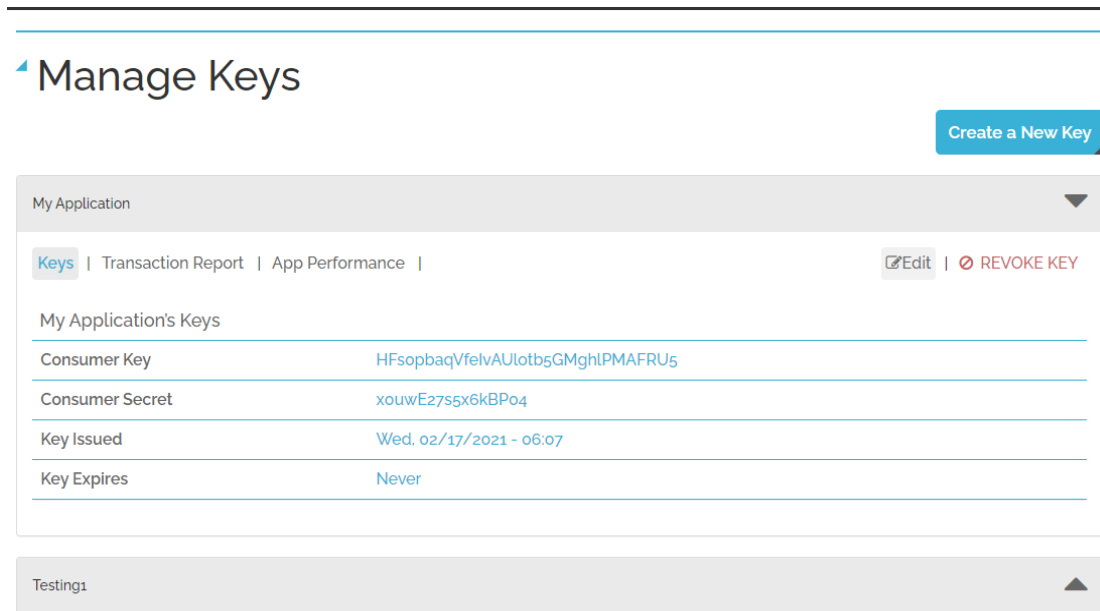


Figure 4.3.1 Map API

For ease of the users, customization of route experience is preferable including quickest driving time, shortest distance or approximate walking distance. In order to accomplish the task, we used *Optimized Route* function and it provides an optimal route.

Mapquest Plugin:

- ❑ Leaflet CSS file is included in header section:

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
```

```
integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMA  
S6/keqq/sMZMZ19scR4PsZChSR7A==" crossorigin=""/>
```

- ❑ After that, Leaflet JavaScript file is included:

```
<!-- Make sure you put this AFTER Leaflet's CSS -->  
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"  
integrity="sha512-XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy2  
0w0vlaXaVUearIOBhiXZ5V3ynxwA==" crossorigin=""></script>
```

- ❑ A div tag is created for map:

```
<div id="map"></div>
```

- ❑ Adding styles including the height of map:

```
#map { height: 180px; }
```

CHAPTER 5: SYSTEM DESIGN AND IMPLEMENTATION

5.1 K-MEANS CLUSTERING

Kmeans is an unsupervised machine learning algorithm that can solve the clustering problems. It is used to identify the clusters of data objects in a dataset which ensures the maximum similarity of objects within each cluster. It was first developed by Macqueen, 1967.^[12]

More formally, K-means clustering is an iterative technique which is used for grouping the similar objects cluster from the dataset and creating K pre-defined non-overlapping clusters. Each object belongs to only one cluster.

5.1.1 K-MEANS CLUSTERING METHOD

Initially, K is defined. The following steps can be performed:

1. Make K cluster centroids randomly.
2. Compute the distance from each object to each cluster centroid
3. Assign each object to a cluster from which cluster centroid distance is minimum
4. Compute the K cluster centroids for current clusters using mean point
5. If K cluster centroids remain the same for two or more consecutive times then stop the procedure, otherwise go to step 3.

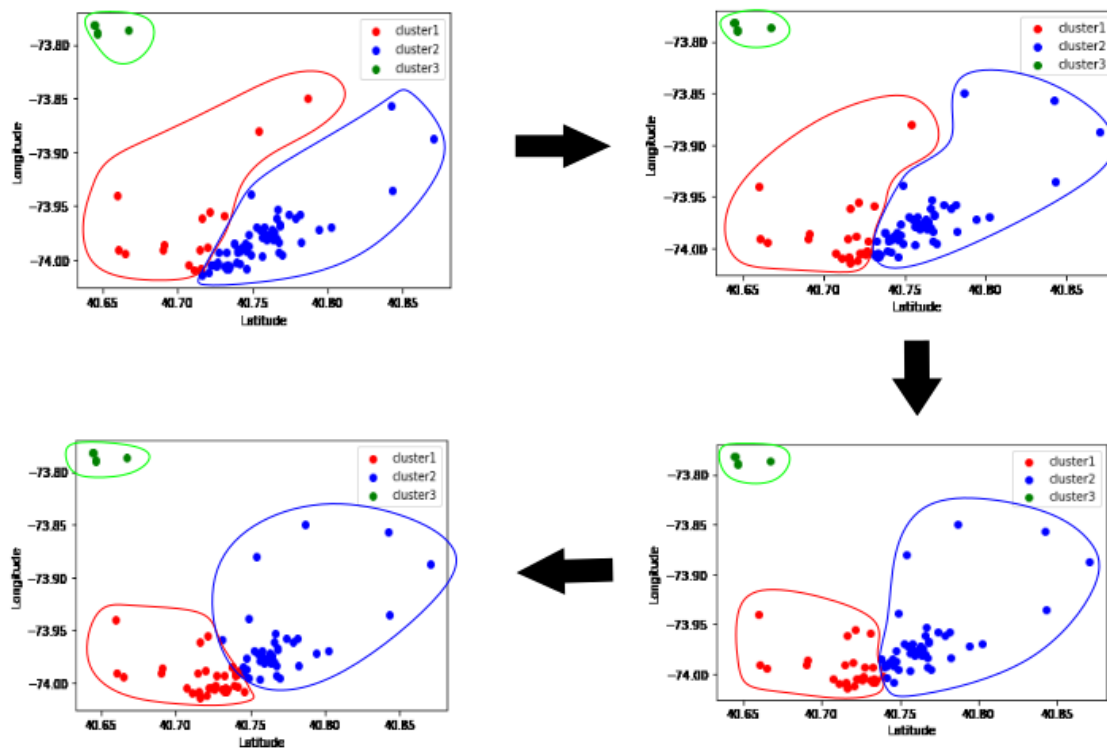


Figure 5.1.1.1 K-means clustering process

5.1.2 HOW TO CHOOSE K

The *Elbow method* is the most popular way to find the optimal K(number of clusters). It works with total variations within each cluster. Using this method, we get an optimal K for which total variations within each cluster is minimum as possible.

To calculate the distance between data points and cluster centroids, we can use any method such as Euclidean distance or Manhattan distance.

5.1.3 K-MEANS CLUSTER IN PYTHON LIBRARY

5.1.3.1 Function Declaration

```
class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init=10, max_iter=300,
tol=0.001, precompute_distances='deprecated', verbose=0, random_state=None, copy_x=True,
n_jobs='deprecated', algorithm='auto')
```

5.1.3.2 Function Parameters

n_clusters(int), default=8: The number of clusters.

init{'k-means++', 'random'}, callable or array-like of shape (n_clusters, n_features), default='k-means++':

- ***'k-means++'***: Initially selects the cluster centers for k-mean clustering.
- ***'random'***: Choose initial *n_cluster* centroids randomly from the dataset.

Function shape (*n_clusters*, *n_features*) returns the initial cluster centers if an array is passed.

It takes arguments (*X*, *n_clusters*, a random state) when a callable is passed.

n_init(int), default=10: Number of times the K-means algorithm will be run.

max_iter(int), default=300: Number of iterations of the K-means algorithm.

tol(float), default=1e⁻⁴: It indicates the relative tolerance regards to Frobenius norm.

precompute_distances{‘auto’, True, False}, default=‘auto’:

- ‘auto’: It does not precompute distances when $n_samples * n_clusters > 12$ million.
- True: always precompute distances.
- False: never precompute distances.

verbose(int), default=0: Verbosity mode.

random_state(int), RandomState instance or None, default=None: Generate random numbers for centroid initialization.

copy_x(bool), default=True: Centering the data points, pre-computing distances seems more accurate. The original data is not altered when copy_x is True, otherwise data is altered. If copy_x is False, before the function returns it is put in back, but few numerical differences can be occurred by subtracting and adding the mean of data.

n_jobs(int), default=None: The number of OpenMP threads which is used for the computation.

algorithm{“auto”, “full”, “elkan”}, default=“auto”: K-means algorithm to use. The classical EM-style algorithm is “full”. Using the triangle inequality the “elkan” variation is more efficient. However it needs extra memory space due to an extra array of shapes (*n_samples*, *n_clusters*) allocation.

5.1.3.3 Function Attributes

cluster_centers_ndarray of shape (*n_clusters*, *n_features*): It returns coordinates of cluster centers. If the algorithm stops before completely converging, it will be inconsistent with *labels_*.

labels_ndarray of shape (*n_samples*,): Indicated labels of each point

inertia_float: It computes the squared distances sum of objects to their closest cluster center.

n_iter_int: Number of iterations run.

5.2 R-TREE

R-tree is a spatial data structure which is used for indexing multi-dimensional information. It is highly useful for spatial data queries, searching and storage. It was proposed by Antonin Guttman in 1984.^[9]

5.2.1 PROPERTIES OF R-TREE

- It consists of a root, internal nodes and leaf nodes.
- Root holds the index of the largest region.
- If the complete overlapping occurs between parent and child nodes, then the parent node holds the index of that child node.
- Minimum bounding region(minimal area of surrounding the region) of current objects is stored into leaf nodes

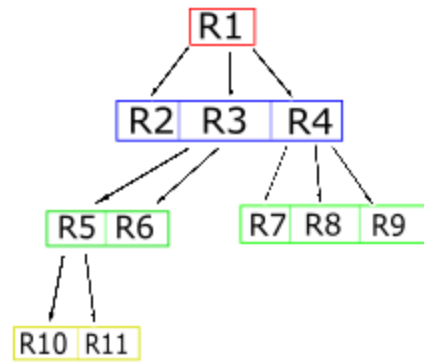
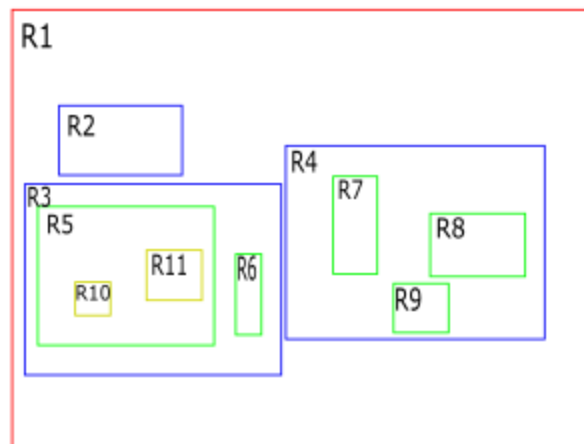


Figure 5.2.1.1 R-tree

5.2.2 R-TREE IN PYTHON LIBRARY

Installation: *pip install rtree*

Importing an index: *from rtree import index*

Constructing an index: *idx = index.Index()*

Insert records into the index: *idx.insert(id, (left, bottom, right, top))*. Insert rectangle (*left, bottom, right, top*) to index *id*.

Intersection Query: *list(idx.intersection((left, bottom, right, top)))*. Returns all the points inside the rectangle (*left, bottom, right, top*).

Nearest point Query: *list(idx.nearest((left, bottom, right, top), n))*. Returns *n* nearest points of the rectangle (*left, bottom, right, top*).

5.3 DESIGN

Step 1: After cleaning and merging the data, the total dataset is classified into 48 (Hour = 24 x Holiday = 2) classes. Each class is represented by a dataframe.

Step 2: For each class, K-means clustering algorithm is applied by setting $K = 10\%$ of total pickup points ($K = \text{Number of clusters}$). We will call each cluster a 'pickup station'. The size of the cluster is the priority of the pickup station.

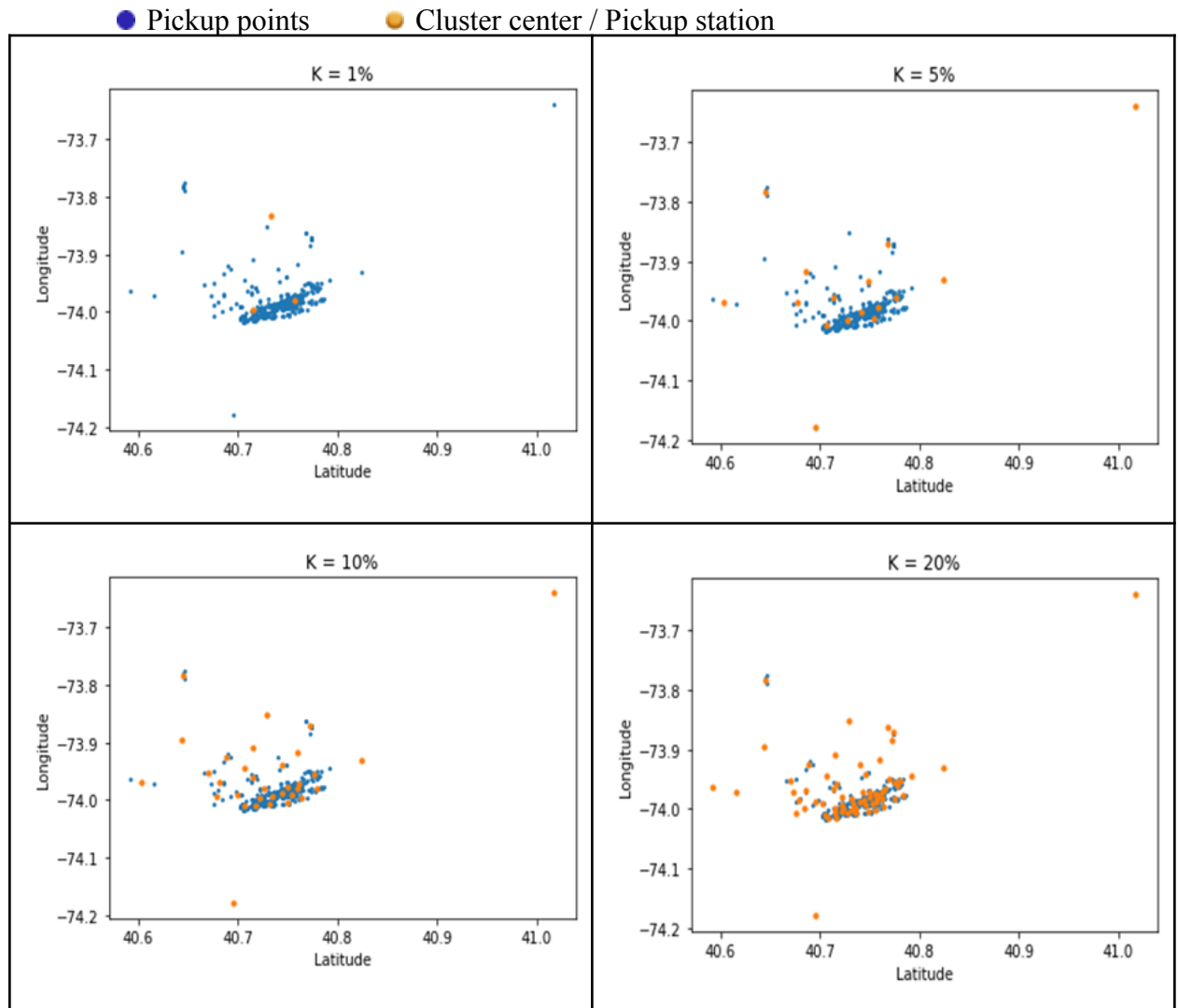


Figure 5.3.1 Comparison of different number of clusters.
(K=number of clusters)

Step 3: For each class, pickup stations are inserted into R-tree.

Step 4: In the frontend user provides three inputs. Location by using the interactive map. Hour by selecting a drop down menu and Holiday by selecting radio button.

Step 5: From *Hour=user provided* and *Holiday=user provided* class we take the closest 50 pickup station using R-tree.

Step 6: The score of each pickup station is calculated by $(1 / \text{distance between user's location and pickup station}) * \text{priority of the pickup station}$.

Step 7: Pickup station with the highest score is selected as the destination.

Step 8: A step by step guide from user's location to destination is displayed in the map using the map API.

CHAPTER 6: RESULT ANALYSIS

6.1 OUTPUT

6.1.1 INITIAL MAP

In the initial map we added cluster region and cluster size to give a better understanding of the data to the user.

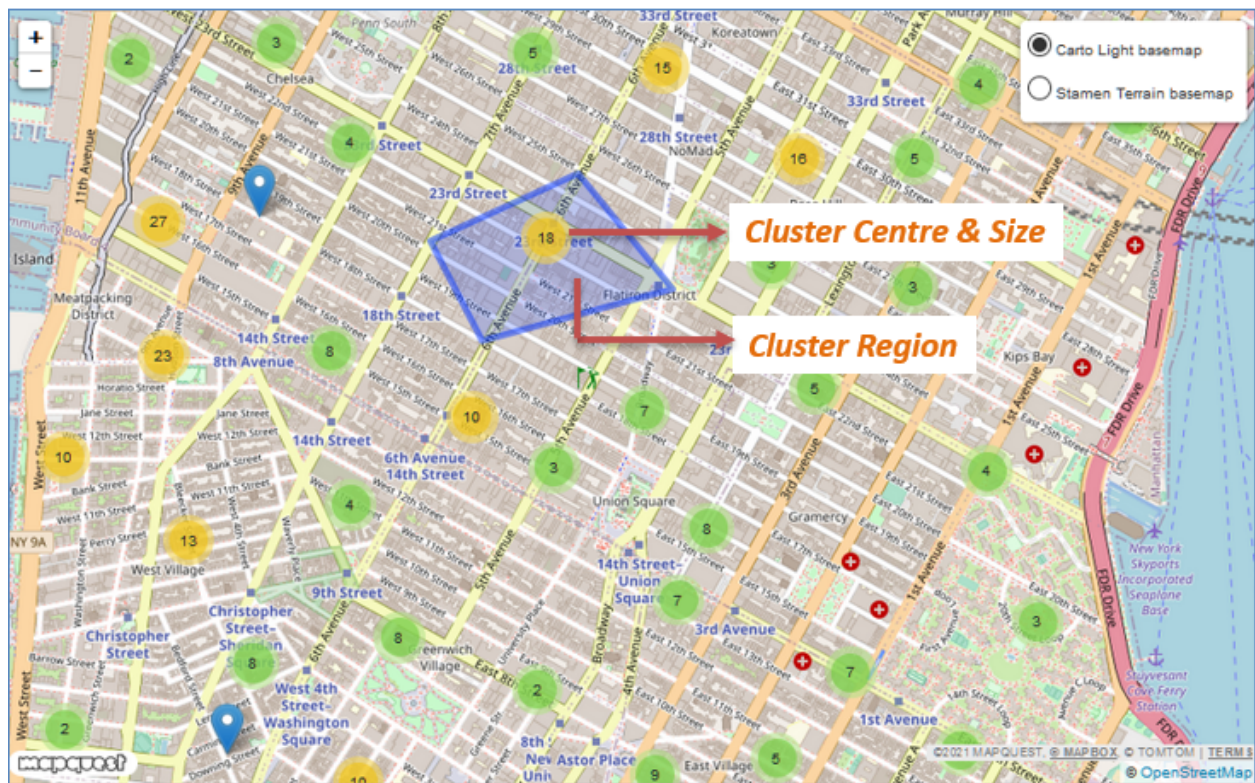


Figure 6.1.1.1 Cluster center and cluster region (Snapshot of our System)

6.1.2 PATH DIRECTION

Here an optimal destination is calculated from the user provided location. A shortest path to destination is also displayed. In the top right of the map we can find the path summary and in the bottom left we can see a detailed guide of how to reach the destination.

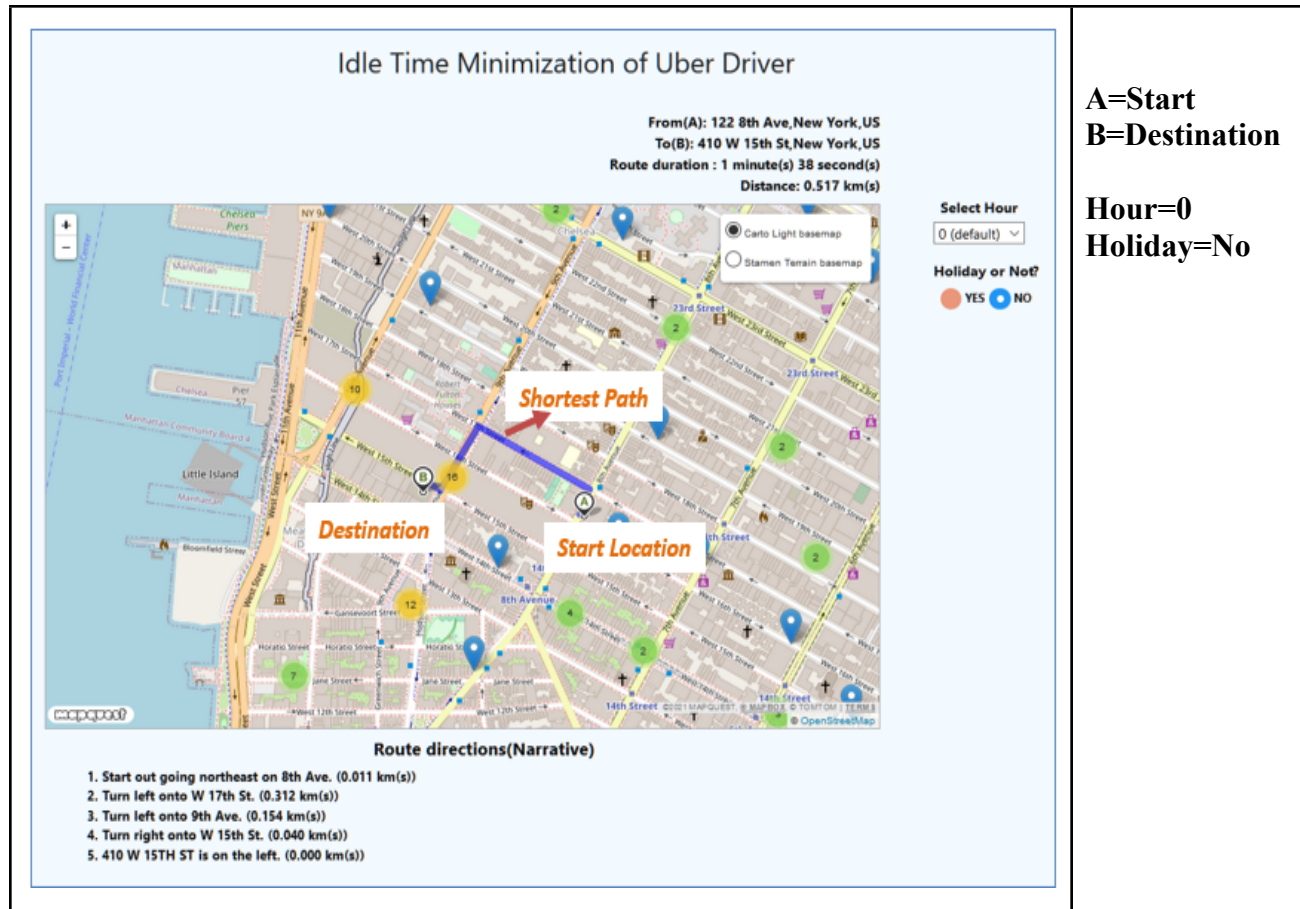


Figure 6.1.2.1 Route-Map view of our System

6.1.3 HOUR INPUT

We provided a drop down menu to select Hour between 0-23.

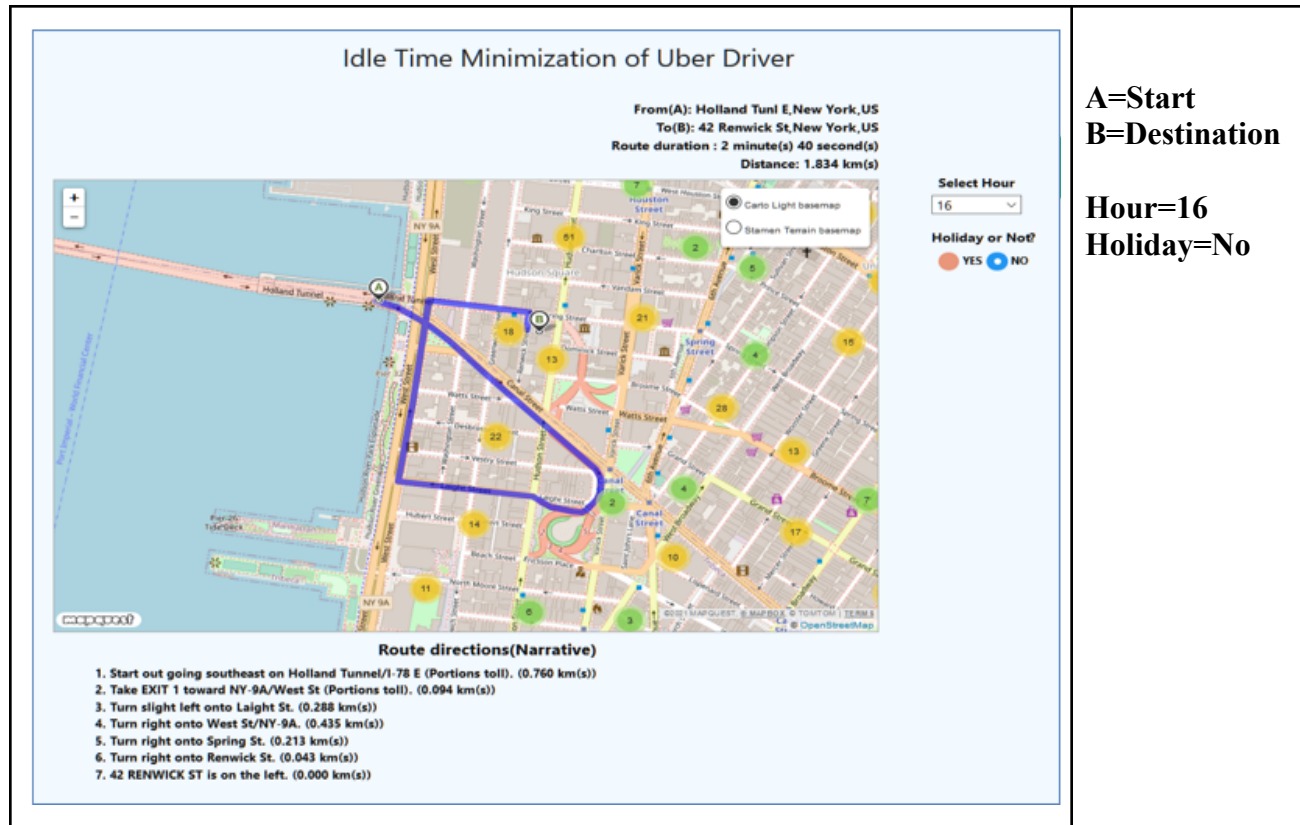


Figure 6.1.3.1 Route-Map view (Hour = Selected by dropdown menu)

6.1.4 HOLIDAY INPUT

We provided a radio button to select the holiday.

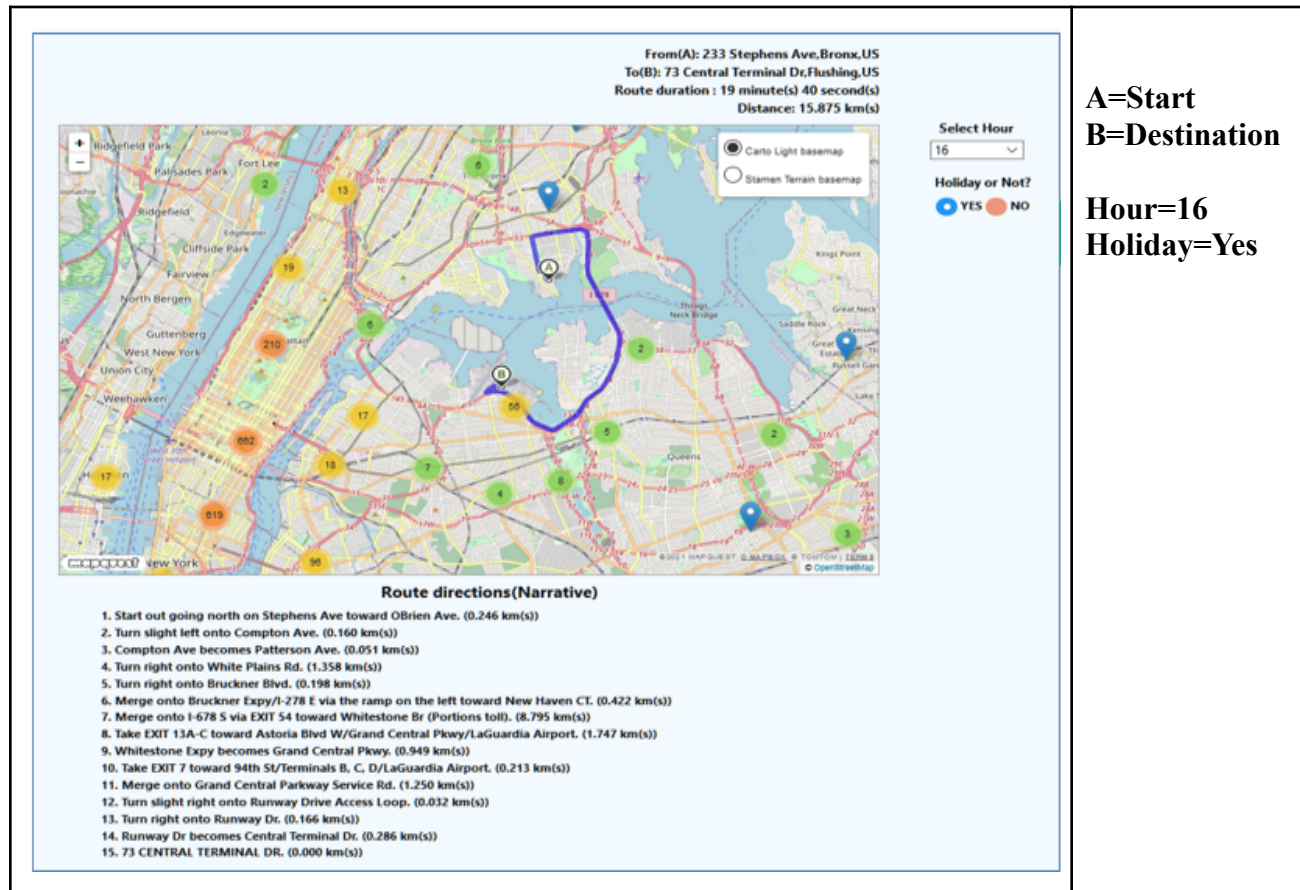


Figure 6.1.4.1 Route-Map view (Holiday = Selected by radio-button feature)

6.1.5 STAMEN TERRAIN BASEMAP

There is also another option to select between Stamen Terrain Basemap and Carto Light Basemap. Stamen Terrain Basemap is an additional feature on map for background customizations which indicates the hill shading and natural vegetation colors. It showcases the advanced labeling and linework generation of dual-carriageway roads.

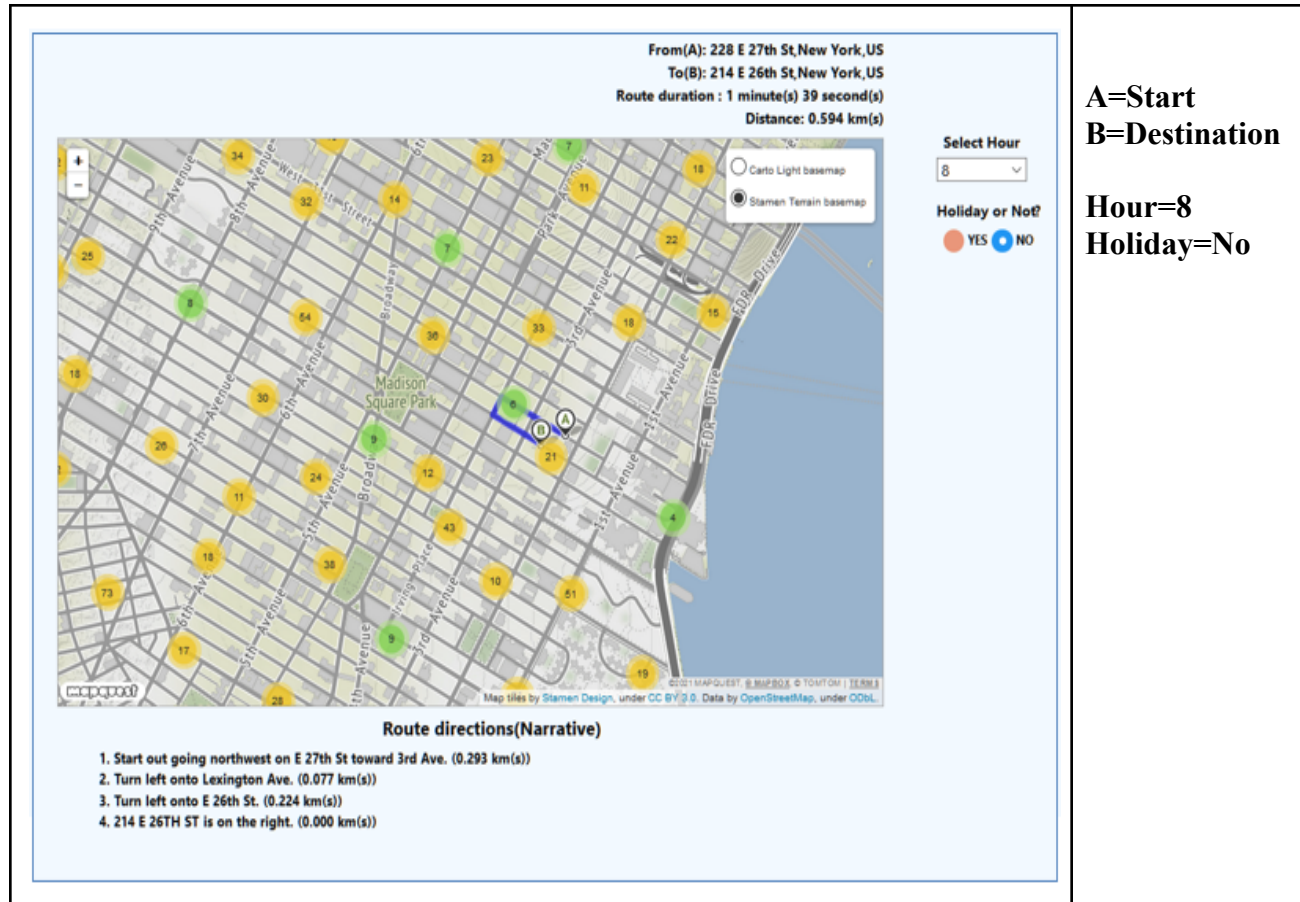


Figure 6.1.5.1 Route-Map view (Stamen Terrain Basemap)

6.2 ACCURACY OF OUR SYSTEM

In our algorithm we only consider a certain amount of closest pickup stations from the driver's current location. Let this amount be T. For example, if T=20, we find the closest 20 pickup stations from the driver's current location and calculate score for each pickup station using the formula $(1 / \text{distance between driver's current location and pickup station}) * \text{priority of that pickup station}$. The best scored pickup station is selected as the optimal destination.

Here in the below table, we have calculated accuracy for different values of T. For a query, first we calculate the optimal station by considering only T closest station, and then we calculate the optimal station by considering every pickup station. If both returns the same optimal station then we have the correct answer for that query.

Below table is generated by taking 120000 pickup points and 10000 randomly generated queries.

Closest pickup stations T	Accuracy	Time(Milliseconds)
T=1	1.404%	441
T=2	26.55%	492
T=10	53.65%	692
T=20	93.92%	873
T=50	99.51%	1122
T=100	99.58%	3089
T=200	99.60%	9941

Table 6.2.1 Accuracy and Time Complexity for different amounts of closest stations(T)

From the table, we can see that, if T increases the accuracy of the system also increases, but so does time complexity. We want our system to be as accurate as possible, but we also want to answer queries as fast as possible.

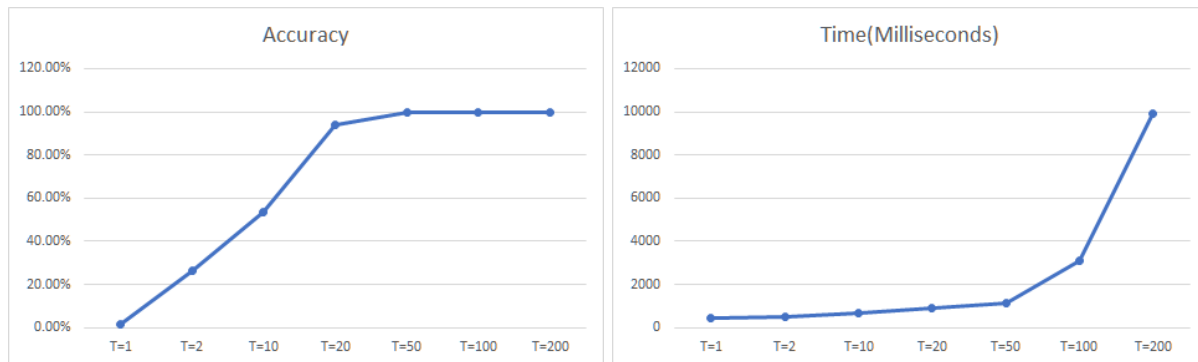


Figure 6.2.1 Accuracy and Time Complexity for different amounts of closest stations(T)

For T=50 we have 99.5% accuracy and it takes a reasonable amount of time to answer the query. So we have selected T=50 for our project.

CHAPTER 7: CONCLUSION

7.1 CONCLUSION

Based on this web application, it brings great value to an Uber Driver in terms of minimizing the idle times along with maximized revenue. We used Uber historic dataset and built K-means cluster models using python scikit-learn library. We find the closer stations from the user's coordinate using a spatial data structure, Rtree.

Then we suggest an optimal station which will minimize the idle times of a driver and maximize the daily revenue as well.

7.2 FUTURE WORK

We can add weather attributes to make our prediction more accurate. Weather data variables such as snow, rain and humidity would make it more meaningful and realistic through filtering.

We can work with different types of Uber cars (UberXL, SUVs, Bike). Specifying the types of Uber car, it would better improve the quality of our prediction model.

REFERENCES

[1]	Investor Presentation 2020 Q4. <i>url:</i> https://s23.q4cdn.com/407969754/files/doc_financials/2019/sr/InvestorPresentation_2020_Feb13.pdf
[2]	Gelly, B. (2018, May 08). Uber Optimization: Finding Passengers Faster. <i>url:</i> https://nycdatasience.com/blog/student-works/uber-optimization-finding-passengers-faster/
[3]	Taylor, L. (2019, March 12). Uber/Lyft Maximization: More Money for The Time. <i>url:</i> https://nycdatasience.com/blog/student-works/uber-lyft-maximization-more-money-for-the-time/
[4]	Jin, Kun, et al. "Reinforcement Learning for Optimizing Driving Policies on Cruising Taxis Services." <i>Sustainability</i> 12.21 (2020)
[5]	Chaudhari, Harshal A., John W. Byers, and Evimaria Terzi. "Putting data in the driver's seat: Optimizing earnings for on-demand ride-hailing." <i>Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining</i> . 2018.
[6]	Ji, Shenggong, et al. "Spatio-temporal feature fusion for dynamic taxi route recommendation via deep reinforcement learning." <i>Knowledge-Based Systems</i> 205 (2020): 106302.
[7]	Qu, Meng, et al. "A cost-effective recommender system for taxi drivers." <i>Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining</i> . 2014.
[8]	Garg, Nandani, and Sayan Ranu. "Route recommendations for idle taxi drivers: Find me the shortest route to a customer!." <i>Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining</i> . 2018.
[9]	Guttman, Antonin. "R-trees: A dynamic index structure for spatial searching." <i>Proceedings of the 1984 ACM SIGMOD international conference on Management of data</i> . 1984.
[10]	Kaggle. <i>url:</i> https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city?select=uber-raw-data-may14.csv
[11]	Data World. <i>url:</i> https://data.world/cegoz22/dimdate
[12]	MacQueen, J. B. (1967). "Some methods for classification and analysis of multivariate

	observations”. <i>Proceedings of the fifth Berkeley symposium on mathematical statistics and probability</i> (Vol. 1, pp. 281–297). California: University of California Press.
--	---