Traffic Condition Recognition Using The K-Means Clustering Method

Project Report

Submitted by

Amrutha S Kumar

Reg No: FIT16MCA-D6

Submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications

of

APJ Abdul Kalam Technological University

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)

ANGAMALY - 683577, ERNAKULAM (DIST.)

MAY 2018

DECLARATION

I hereby declare that the report of this project work, submitted to the Department of Computer Application, Federal Institute of Science and Technology (FISAT), Angamaly in partial fulfillment of the award of the degree of Master of Computer Applications is an authentic record of my original work.

The report has not been submitted for the award of any degree of this university or any other university.

Date :

Place: Amrutha S Kumar

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)

ANGAMALY, ERNAKULAM-683577

CERTIFICATE

This is to certify that the project report titled " ***Traffic Condition Recognition Using The K-Means Clustering Method****" submitted by Amrutha S kumar (Reg No. FIT16MCA-D6) towards partial fulfillment of the requirements for the award of the degree of Master of Computer Application is a record of bonafide work carried out by her during the year 2018.*

Project Guide

Head of the Department

Submitted for the viva-voce held on ..... at .....

Examiners :

ACKNOWLEDGEMENT

I am extremely glad to present my main project as a part of our curriculum. I take this opportunity to express my sincere thanks to those who helped me in bringing out the report of my project.

I am deeply grateful to **Mr. George Issac** , Principal, FISAT, Angamaly for his help and suggestions throughout the project and I express my sincere thanks to **Dr. C. Sheela** ,Vice Principal FISAT, Angamaly.

My sincere thanks to **Mr. Santhosh Kottam**, HOD, Department of Computer Applications, FISAT, who had been a source of inspiration. I express heartiest thanks to **Dr. Sreeraj M**, Project Scrum Master for his encouragement and valuable suggestion. I express my sincere gratitude to **Ms.Joice T**, my project guide for her valuable support. I express my heartiest gratitude to all the faculty members in our department for their constant encouragement and never ending support throughout the project.

Finally I express my thanks to all my friends who gave me wealth of suggestion for successful completion of this project.

Amrutha S Kumar

ABSTRACT

Prediction of travel time has major concern in the research domain of Intelligent Transportation Systems (ITS). Clustering strategy can be used as a powerful tool of discovering hidden knowledge that can easily be applied on historical traffic data to predict accurate travel time. In our Modified K-means Clustering (MKC) approach, a set of historical data is portioned into a group of meaningful sub- classes (also known as clusters) based on travel time, frequency of travel time and velocity for a specific road segment and time group. The information from these are processed and provided back to the travellers in real time. Traffic flow modelling and driving condition analysis have many applications to various areas, such as Intelligent Transportation Systems (ITS), adaptive cruise control, pollutant emissions dispersion and safety.

K-means clustering is a type of unsupervised learning, which is used when you have unlabelled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided.

The project entitled "Traffic Condition Recognition Using The K-Means Clustering Method" The use of a machine learning technique, namely K-means, for the Traffic analysis. In this project we used the K-means al- gorithm for clustering the dataset. In this project impart some libraries K-means, folium, and Xgboost .

Contents

Chapter 1

Introduction

1.1 Introduction

Recently, accurate estimation of travel times has been central for traffic data analysis to various Advanced Travelers Information System (ATIS) and ITS applications such as trip planning, vehicular navigation systems and dynamic route guidance systems. Moreover, Travel time prediction is also becoming increasingly important with the development of ATIS. In addition, Travel time forecasting provides information that helps travellers to decide whether they should change their routes, travel mode, starting time or even cancel their trip. So, the reliable and accurate travel time prediction on road network plays an important role in any kind of dynamic route guidance systems to fulfil the users' desires. On top of that, the importance of travel time information is also indispensable to find the fastest path (i.e. shortest path

according to travel time) that connects the origin and destination. Besides, accurate travel time information is delivered to industries in progress their service quality by delivery on time. Travel time prediction is based on vehicle speed, traffic flow and occupancy which are extremely sensitive to external event like weather condition and traffic incident. The traffic flow of a specific road network fluctuates based on daily, weekly and occasional events. For example, the traffic condition of weekend may differ from that of weekday. So, time varying feature of traffic flow is one of the major issues to estimate accurate travel time . In this study, we focus a new method that is able to predict travel time reliably and accurately .

Accurate estimation of travel times is the key factor in traffic data analysis. The traffic conditions may change every time according to different factors like office time, weekends, weather, accidents etc. It is important to know how much travel time it would take to reach the destination and what is the shortest way to reach there. This will help the traveler to check and determine the shortest time path while they are booking the online taxi for the trip. Without data analysis and live update the estimation and identification of shortest travel time at a time is very difficult. Hence in such situations the challenge is to analyze the historic data and other traffic conditions and provide the live update to traveler and driver to determine the shortest time. This information will help travelers to decide whether they should change their routes, travel mode, starting time or even cancel their trip.

Chapter 2

Proof of Concept

2.1 Proof of Concept

The project called "Traffic Condition Recognition Using The K-Means Clustering Method" is based on [1]. This paper has used different technologies for implementing this. This paper presents a methodological approach to traffic condition recognition, based on driving segment clustering. This study focuses on the application of driving condition recognition. For this purpose, driving features are identified and used for driving segment clustering, using the k-means clustering algorithm.

Travel time prediction is based on vehicle speed, traffic flow and occupancy which are extremely sensitive to external event like weather condition and traffic incident.

Proposed MKC method is able to address the arbitrary route on road networks that is given by user.

2.2 Objectives

The main objective of this project is to find the shortest travel time by analyzing the various historic data and provide the real time update back to the user while they are booking an online taxi for the trip. Travel time forecasting provides information that helps travelers to decide whether they should change their routes, travel mode, starting time or even cancel their trip. This project aims to use the historic data such as frequency of travel time and velocity for a specific road segment, traffic details during different days etc. With the use of an unsupervised machine learning technique, namely K-means Clustering the historic data will be analyzed and grouped in to various groups according to the feature similarity. This groups will be used for grouping the new data and also for predicting the shortest travel time. By using this technique the algorithm becomes strong and able to provide the accurate travel time for a trip at real time to traveler.

Chapter 3

Implementation

3.1 Implementation

All operations are performed in Python and libraries used are numpy, for mathematical operations, pandas for handling data, scipy for scientific computing and technical computing, folium for manipulate data in python and visualize it in on a leaflet map via folium, Xgboost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning and seaborn for statistical data visualization.

The competition dataset is based on the 2016 NYC Yellow Cab trip record data made available in Big Query on Google Cloud Platform. The data was originally published by the NYC Taxi and Limousine Commission (TLC). The data

was sampled and cleaned for the purposes of this playground competition. Based on individual trip attributes, participants should predict the duration of each trip in the test set.

Folium makes it easy to visualize data that's been manipulated in Python on an interactive Leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing Vincent/Vega visualizations as markers on the map. The library has a number of built-in tilesets from Open Street Map, Mapbox, and Stamen, and supports custom tile sets with Mapbox or Cloud made API keys. Folium supports both GeoJSON and TopoJSON over lays, as well as the binding of data to those overlays to create choropleth maps with color –brewer color schemes.

Define the Feature Eng in train then create new features for the date or time. Then calculate Distance and map in the program distance finding we use haversine function and manhattan function. Using K-means algorithm clustering the location and Feature engineering. Both train and test the dataset. Visualization contains histogram, count plot etc. Calculate the distance and speed in hour and minute Map the location of all pickups using Folium. Create a simple leasllt map. Exclude feature not for model development. Start training the XGB - eXtreme Gradient Boosting Note, the parameters should be further tuned. Examine features Use Cv to determined number of rounds.

Numerous researchers have paid their attention on the accurate travel time prediction as it is one of the major issues for effective dynamic route guidance systems. Various methodologies have been investigated till date for computation and prediction of travel times with varying degree of successes. In this section, a historical background on the topic of travel time prediction is discussed briefly. We propose MKC method. In this study, we try to eliminate the shortcomings of traditional K-means Clustering approach. he key challenges of this research are to reduce prediction error as well as to predict the uncertain situation. At the same time, proposed method can also be scalable to large network with arbitrary travel routes.

In this section, a new method for foretelling travel time from historical traffic data using MKC method is depicted. Cluster analysis or clustering is an assignment of separating the set of observations into subset. A cluster is therefore a collection of objects which are similar between themselves and are dissimilar to the objects belonging to other clusters. K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problems. The main advantages of K-means algorithm are its simplicity and speed which allows it to run on large datasets. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume K clusters) fixed a priori. The main concept is to define K centroids, one for each cluster. These centroids should be positioned in a cunning way because location verification causes result verification. So, the better choice is to place them as much as possible far away from each other. In our MKC method, we incorporate a technique so that centroids of different clusters

maintain a sufficient difference. The main disadvantage of K-means clustering is that it doesn't yield the same result with each run, since the resulting clusters depend on the initial random assignments. We have formulated MKC method in a cunning way such that we can eliminate this kind of shortcoming of well-defined K-mean initially, an origin with start time and destination is initialized by user. A route may consist of several road segments from origin to destination. First of all, we apply our MKC method on the data set of the first road segment to calculate the end time of first road segment which in turn becomes the start time of the next road segment. Finally, applying successive repetition approximate travel time from origin to destination can measured.

3.2 DataSet Used

• New York City Taxi Trip Duration Share code and data to improve ride time predictions

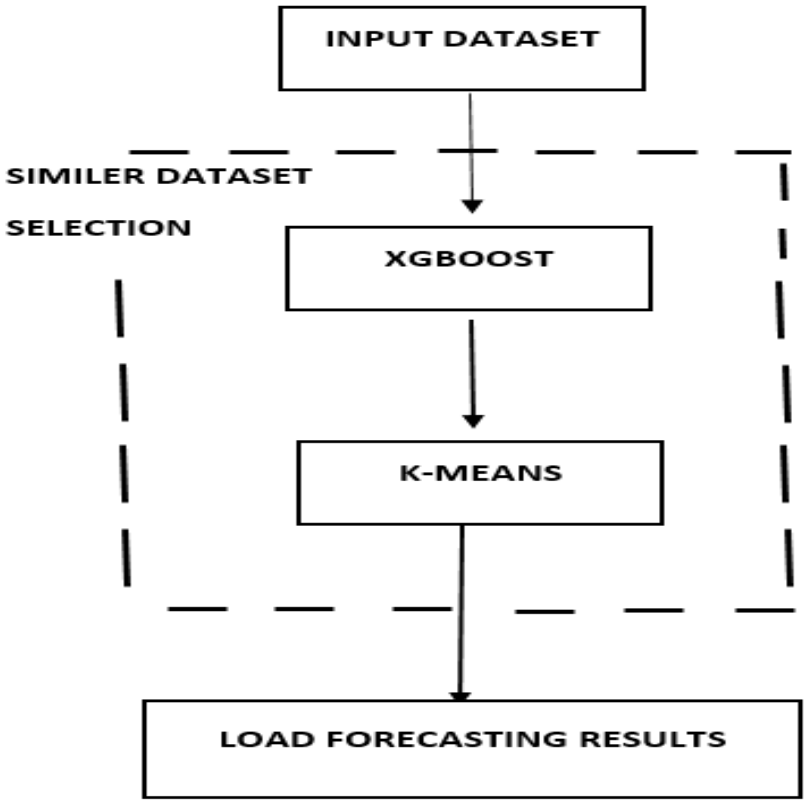
• Source:https://www.kaggle.com/c/nyc-taxi-trip-duration

• Size: The dataset contains in total 3 files.

- train.csv - the training set (contains 1458644 trip records)
- test.csv - the testing set (contains 625134 trip records)
- sample-submission.csv - a sample submission file in the correct format

3.3 System Architecture

3.3.1 Block Diagram



The block diagram that describes the operation of the system .The input to the system is a dataset, and the outputs are load images . In the data analysis step in the dataset is used K-Means algorithm.

3.3.2 Algorithm

Algorithm 1 Traffic Condition Recognition Using The K-Means Clustering Method
Input : Load the csv file as dataset
Output : Load Images and Map

Step1 : Start

Step2 : Read the dataset.

Step3 : Clustering the dataset using K-Means algorithm.

Step4 : XGBoost framework using boosting the dataset.

Step5 : Map polted using folium Interactive maps .

Step6 : Load the output images on a folder.

Step7 : Stop .

3.4 Algorithms Used

3.4.1 K-Means

Clustering is a type of unsupervised learning. This is very often used when you don't have labeled data. K-Means Clustering is one of the popular clustering algorithm. The goal of this algorithm is to find groups (clusters) in the given data. In this post we will implement K-Means algorithm using Python from scratch. K-Means is a very simple algorithm which clusters the data into K number of clusters. The following image from pypr is an example of K-Means Clustering.

K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centroids, one for each cluster. These centroids shoud be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes

are done. In other words centroids do not move any more. Finally, this algorithm aims at minimizing an objective function, in this case a squared error function.

3.4.2 XGBoost

XGBoost is an open-source software library which provides the gradient boosting framework for C++, Java, Python, R, and Julia. It works on Linux, Windows, and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". Other than running on a single machine, it also supports the distributed processing frameworks Apache Hadoop, Apache Spark, and Apache Flink. It has gained much popularity and attention recently as it was the algorithm of choice for many winning teams of a number of machine learning competitions.

XGBoost is used for supervised learning problems, where we use the training data (with multiple features) xi to predict a target variable yi. Before we dive into trees, let us start by reviewing the basic elements in supervised learning.As mentioned you can play with the different parameters of the XGBoost algorithm to tweak the model's outcome. So below is a short, but very nice, way of itterating through model parameters to tweak the model. So it's implementation is simple: just uncomment the code and run the kernel.

Basic exploratory data analysis

•Feature engineering

•Build XGB model and generate predictions

•Explore the possibility of network analysis

•Further tune the XGB model parameters

3.5 Issues Faced and Remedies Taken

- **Issue 1 :** Finding out the dataset was a problem.
- **Remedy :** Finally got the dataset from New York City Taxi Trip Duration Share code and data to improve ride time predictions.
- **Issue 2 :** There was as an issue while running the code , showing a ValueError.
- **Remedy :** Installed python and XGBoost

Chapter 4

Result Analysis

4.1 Result Analysis

Statistical analysis is the science of collecting data uncovering patterns and trends. its really just another way of saying "statistics". After collecting data you can analyze it to:

•Summarize the data.

•Create new features for date/time weekday.

•Calculate measures of speed; this tell you if your data is highly clustered or more spread out. The standard deviation is one of the more commonly used measures of spread; it tells you how spread out your data is about mean.

•Map Locations of all pickups

•All outputs saved on a floder, That floder contains Distribution of log(Trip Duration, Heatmap of trip distribution,K-Means clusters of pick-up/drop-off locations,Relationship between time and distance, Feature importance from XGB.

4.1.1 Result

Traffic Condition Recognition and Analysis using the K-Means Clustering XGBoost algorithms which is based on data analysis. Both systems used a three step process; in the first step, Location clustering the dataset using K-Means algorithm, Create new features for date/time weekday. Statstics function are located in the sub pachage 'scipy.stats'. Second step, Find the distance, speed, travel time using xgboost (provides the gradient boosting)In third step, Calculate distance and map, Apply XGBoost model on test dataset, Map the pickup points using folium (Interactive maps). All outputs saved on a floder, That floder contains Distribution of log(Trip Dura- tion),Heatmap of trip distribution,K-Means clusters of pick-up/drop-off locations,Relationship between time and distance, Feature importance from XGB.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

In conclusion, we can infer from this project that by using the machine learning technique K-means Clustering it is able to identify the shortest travel time for a trip by analyzing the historic data such as speed in the road and other traffic conditions. The algorithm getting stronger and provide accurate results time by time by analyzing more data. In can be concluded that the accurate travel time estimation at real time during the taxi booking will help the traveler to plan the trip accordingly. Also with a more detailed and complete dataset, we can perform better and more accurate predictions. These are some prospects of this project that can be explored in this future in order to improve traffic time estimation and make this available to the traveler and also for the driver to make the dynamic route calculations during a

trip according to current traffic situations.

5.2 Future Scope

For future work we can collect live traffic conditions based on dataset also using GPS for live updates. I think that intelligent travel systemstThe increasing number of mega cities and the population growth in developed and developing countries has increased the importance of deploying an intelligent transport system (ITS). ITS system constitutes both, road transport and an efficient metro/underground rail system. ITS involves the revamp of overall technological aspects such as GPS, Carrier Access for Land Mobiles (CALM), Dedicated Short Range Communication (DSRC) etc. Globally, the concerned government departments understand the importance of implementing an efficient ITS system, which is an important driving factor for the market growth. Therefore, these departments are formulating specific programs and taking initiatives to implement the system. For example, the U.S. Department of Transport (DOT) is focusing on intelligent infrastructure, intelligent vehicles and integration of these two factors.

Chapter 6

Appendix

6.1 Screenshot

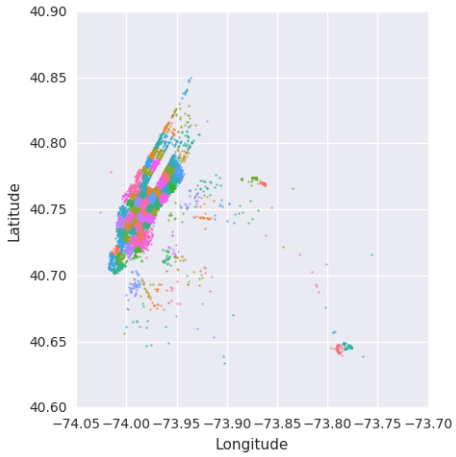


Figure 6.1: K-means clusters of pick-up/drop-off location

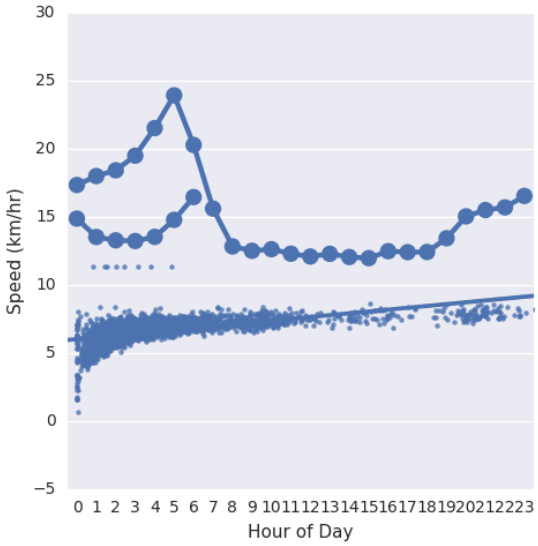


Figure 6.2: Relation between time and distance(Days)

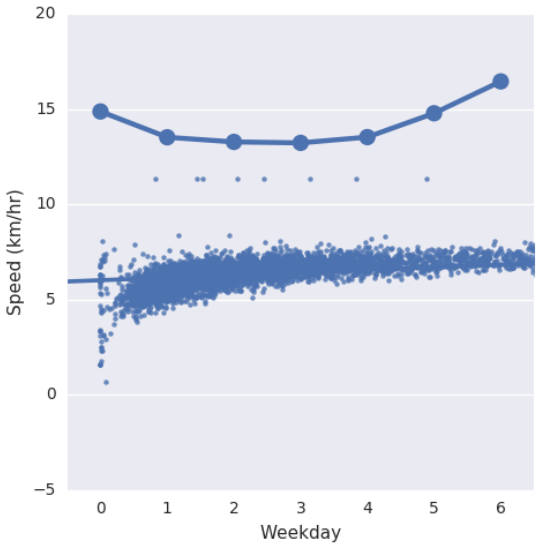


Figure 6.3: Relation between time and distance(Weekday)

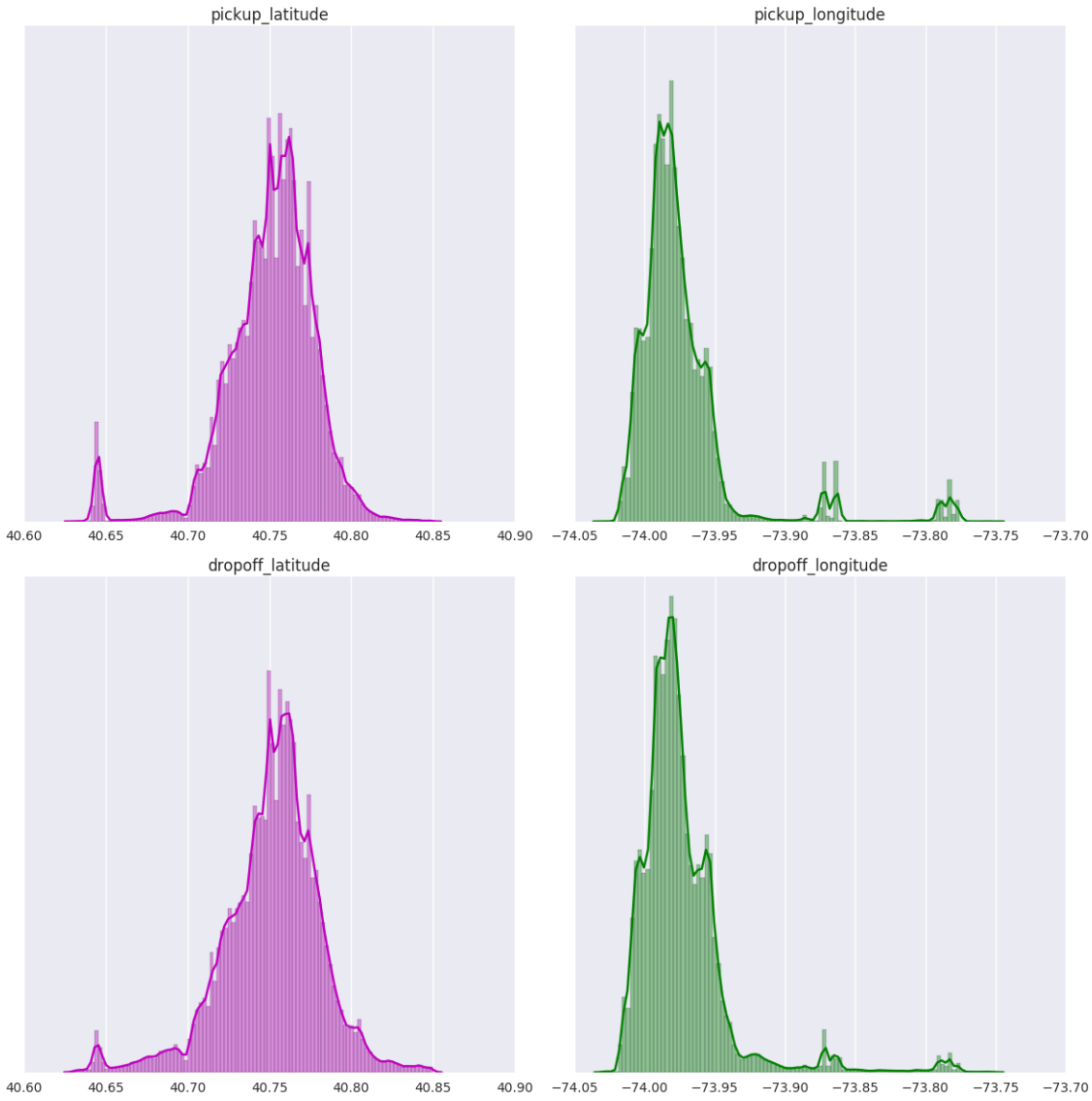


Figure 6.4: Plot latitude and longitude

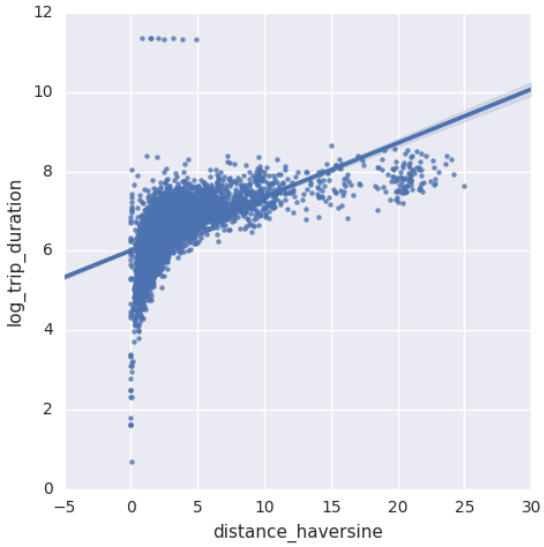


Figure 6.5: Time distance plot

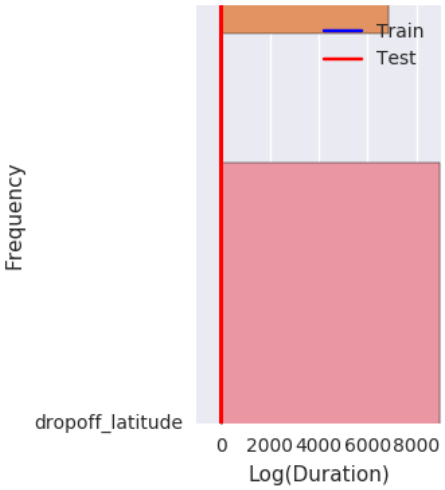


Figure 6.6: Two histogram plot

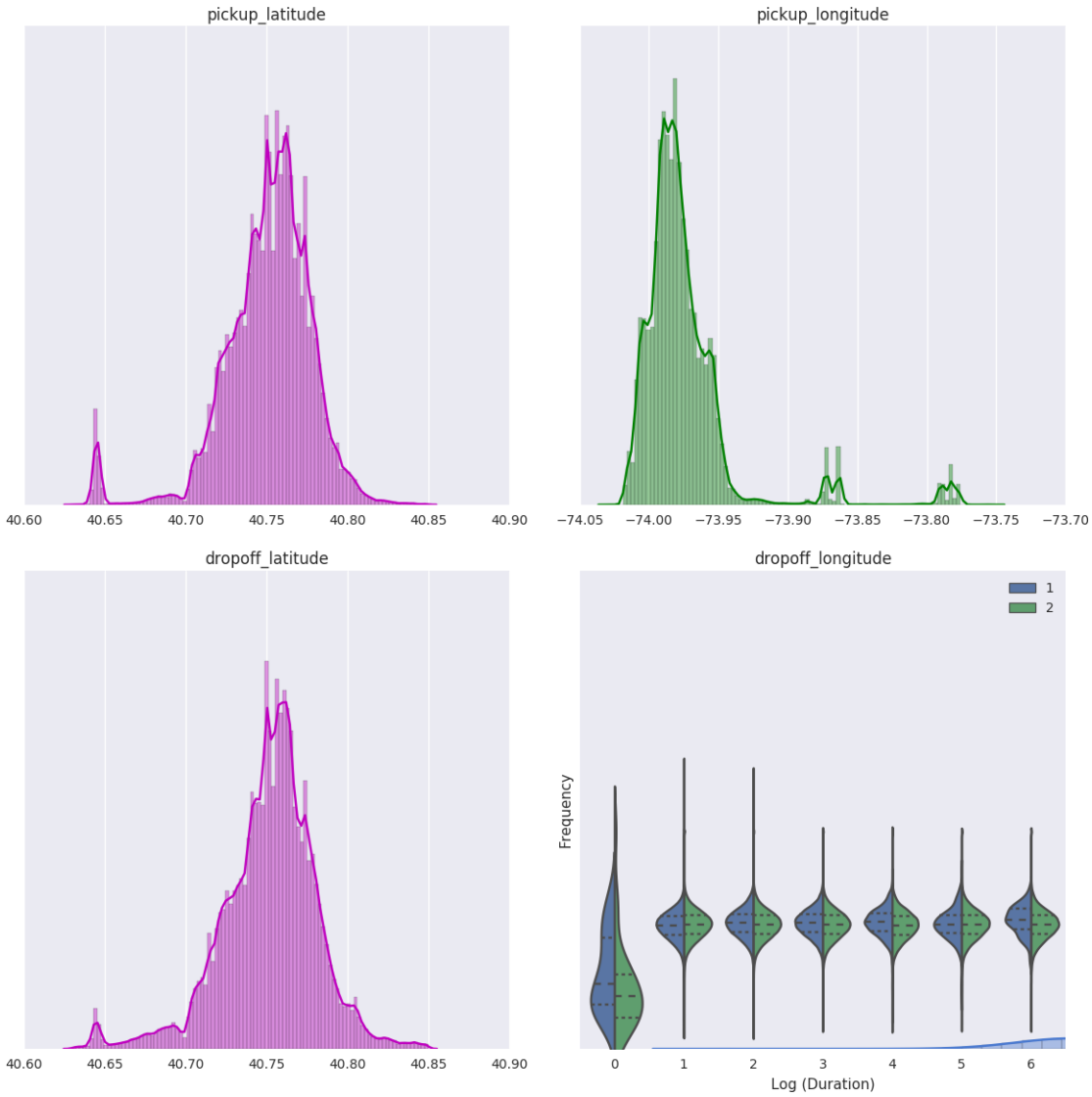


Figure 6.7: Distribution of trip

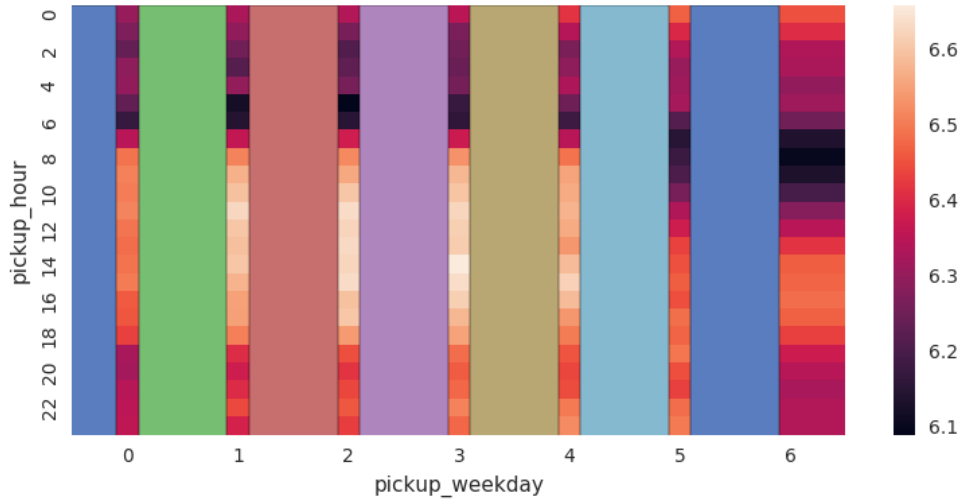


Figure 6.8: Heatmap of trip duration

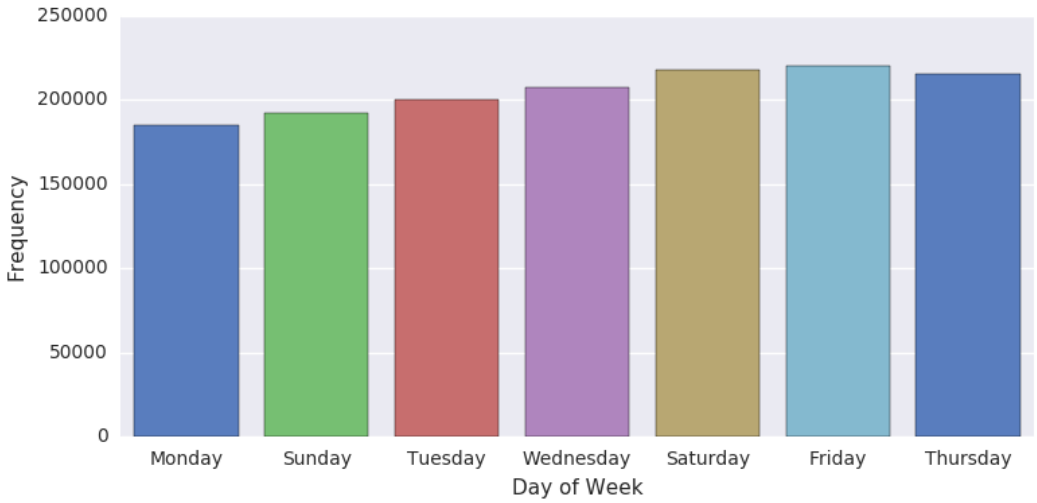


Figure 6.9: Count day of week

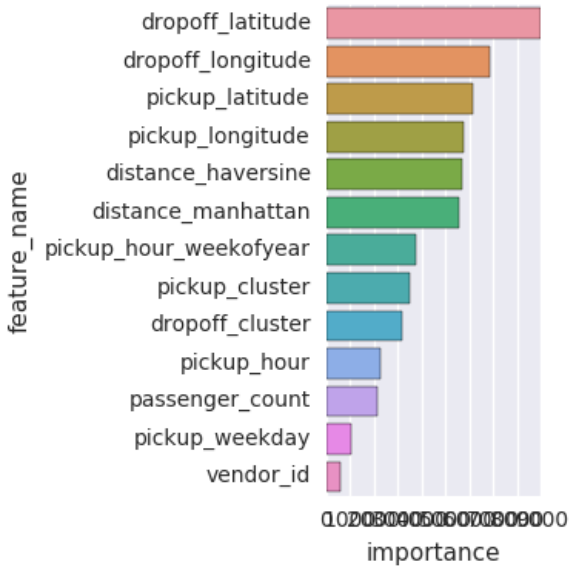


Figure 6.10: Features importance from XGB

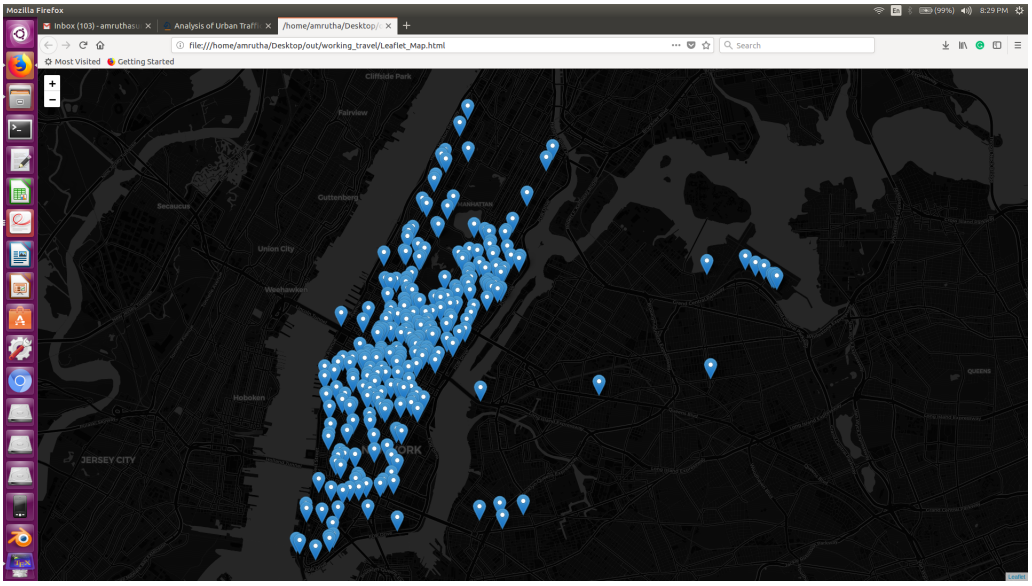


Figure 6.11: Map plat pick up points

6.2 Source Code

6.2.1 Main.py

```
# Import libraries
from modules.plots import *
from modules.map import *
import pandas as pd
import numpy as np
from scipy import stats  #statstics fn are located in the sub pachage scipy.stats
from sklearn.cluster import KMeans
import folium #Interactive maps
import xgboost as xgb #provides the gradient boosting

def FeatureEng(train):
#Create new features for date/time & fig4  weekday
train['pickup_datetime'] = pd.to_datetime(train.pickup_datetime)
train['pickup_weekday_name'] = train['pickup_datetime'].dt.weekday_name
train['pickup_weekday'] = train['pickup_datetime'].dt.weekday
train['pickup_hour_weekofyear'] = train['pickup_datetime'].dt.weekofyear
train['pickup_hour'] = train['pickup_datetime'].dt.hour

#Calculate distance & map
```

```
20      train['distance_haversine'] = haversine_array(train['pickup_latitude
    '].values,
21      train['pickup_longitude'].values,
22      train['dropoff_latitude'].values,
23      train['dropoff_longitude'].values)
24
25      train['distance_manhattan'] = manhattan_distance(train['
    pickup_latitude'].values,
26      train['pickup_longitude'].values,
27      train['dropoff_latitude'].values,
28      train['dropoff_longitude'].values)
29      #Location clustering
30      coords = np.vstack((train[['pickup_latitude', 'pickup_longitude']].
    values,
31      train[['dropoff_latitude', 'dropoff_longitude']].values))
32      sample_ind = np.random.permutation(len(coords))[:5000]
33      kmeans = KMeans(n_clusters=100).fit(coords[sample_ind])
34
35      train['pickup_cluster'] = kmeans.predict(train[['pickup_latitude', '
    pickup_longitude']])
36      train['dropoff_cluster'] = kmeans.predict(train[['dropoff_latitude',
     'dropoff_longitude']])
37
38      # Return dataframe
39      return(train)
40
41      # Read train set
```

```
42    train = pd.read_csv('train.csv')
43    test = pd.read_csv('test.csv')
44
45    # Feature Engineering
46    train['log_trip_duration'] = np.log(train['trip_duration'].values +
      1)
47    city_long_border = (-74.03, -73.75)
48    city_lat_border = (40.63, 40.85)
49    train = train[(train.pickup_latitude > city_lat_border[0]) & (train.
      pickup_latitude < city_lat_border[1])]
50    train = train[(train.dropoff_latitude > city_lat_border[0]) & (train
      .dropoff_latitude < city_lat_border[1])]
51    train = train[(train.pickup_longitude > city_long_border[0]) & (
      train.pickup_longitude < city_long_border[1])]
52    train = train[(train.dropoff_longitude > city_long_border[0]) & (
      train.dropoff_longitude < city_long_border[1])]
53
54    # For both train and test
55    train = FeatureEng(train)
56    test = FeatureEng(test)
57
58    # Visualization
59    LonLatPlot(train)
              #6  Lat_Lon_Plot
60    ViolinPlot(train, 'log_trip_duration','passenger_count', 'vendor_id'
      )
```

```
61    Hist(train['log_trip_duration'], 'Log (Duration)', 'Log Trip
    Duration His') #1   Log Trip Duration His
62    CountPlot(train, 'pickup_weekday_name','Day of Week', 4, 2)
            #2  Day of Week_Count_Plot
63    PivotPlot(train, 'log_trip_duration', 'pickup_hour', 'pickup_weekday
    ')        #7  pickup_hour_and_pickup_weekday_Pivot
64    Time_Distance_Plot(train)
            #8  Time_distance_pivot
65
66    # Calculate distance and speed
67    train['avg_speed_h'] = train['distance_haversine'] / train['
    trip_duration'] * 3600# in the unit of km/hr
68    train['avg_speed_m'] = train['distance_manhattan'] / train['
    trip_duration'] * 3600# in the unit of km/hr
69    SpeedPlot(train,'pickup_weekday','avg_speed_h','Weekday')
70    SpeedPlot(train,'pickup_hour','avg_speed_h','Hour of Day')
71
72    # Map Locations of all pickups
73    MapPlot(train)
74    MapPlot(train[:10000], 'pickup_cluster', legend_plot = False)
75
76    # Create a simple leafllet map
77    locations = train[['pickup_latitude', 'pickup_longitude']]
78    locationlist = locations.values.tolist()[:300]
79    map_object = folium.Map(location=[40.767937,-73.982155], tiles='
    CartoDB dark_matter', zoom_start=12)
80    #marker_cluster = folium.MarkerCluster().add_to(map_object)
```

```
81      for point in range(0, len(locationlist)):
82      folium.Marker(locationlist[point]).add_to(map_object)
83      folium.Map.save(map_object, "Leaflet_Map.html")
84
85      # Exclude feature not for model development
86      feature_names = list(train.columns)
87      do_not_use_for_training = ['id', 'log_trip_duration', '
        pickup_datetime', 'dropoff_datetime', 'trip_duration',
88      'pickup_weekday_name','pickup_date', 'avg_speed_h', 'avg_speed_m','
        store_and_fwd_flag']
89      feature_names = [f for f in train.columns if f not in
        do_not_use_for_training]
90
91      # Examine features
92      feature_stats = pd.DataFrame({'feature': feature_names})
93      feature_stats['train_mean'] = np.nanmean(train[feature_names].values
        , axis=0)
94      feature_stats['train_std'] = np.nanstd(train[feature_names].values,
        axis=0)
95      feature_stats['train_nan'] = np.mean(np.isnan(train[feature_names].
        values), axis=0)
96
97      ## Start training the XGB - eXtreme Gradient Boosting
98      # Note, the parameters should be further tuned
99      xgb_pars = {'eta': 0.3, 'colsample_bytree': 0.3, 'max_depth': 10,
100     'subsample': 0.5, 'lambda': 1, 'booster' : 'gbtree', 'silent':0,
101     'eval_metric': 'rmse', 'objective': 'reg:linear'}
```

```
102     Xtr = train[feature_names]
103     Ytr = train['log_trip_duration']
104     dtrain = xgb.DMatrix(Xtr, label=Ytr) #xgboost dataset
105
106     # Use CV to determind number of rounds
107     max_num_round = 100
108     xgb_cv = xgb.cv(xgb_pars, dtrain, max_num_round, nfold=5,
109     callbacks=[xgb.callback.print_evaluation(show_stdv=True)])
110
111     #Use the best number of rounds from CV to train the model again
112     selected_num_round = 92
113     xgb_model = xgb.train(xgb_pars, dtrain, selected_num_round)
114
115     # Examine feature importance & fig 2
116     feature_importance_dict = xgb_model.get_fscore()
117     feature_importance_table = pd.DataFrame({'feature_name': list(
    feature_importance_dict.keys()),
118     'importance': list(feature_importance_dict.values())})
119     feature_importance_table.sort_values(by ='importance', ascending=
    False, inplace= True)
120     PlotFeatureImp(feature_importance_table)
121
122     # Apply model on test dataset
123     Xte = test[feature_names]
124     dtest = xgb.DMatrix(Xte) #xgboost dataset
125     Yte = xgb_model.predict(dtest)
126     test['log_trip_duration'] = Yte
```

```
127     test['trip_duration'] = np.exp(Yte) - 1
128     test[['id', 'trip_duration']].to_csv('sample_submission.csv', index=
        False)
129     Two_Hist_Plot(train, test)
130
131
132
```

Listing 6.1: code1

6.2.2 map.py

```
# -*- coding: utf-8 -*-
import numpy as np

def haversine_array(lat1, lng1, lat2, lng2):
lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
AVG_EARTH_RADIUS = 6371  # in km
lat = lat2 - lat1
lng = lng2 - lng1
d = np.sin(lat * 0.5) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(lng
  * 0.5) ** 2
h = 2 * AVG_EARTH_RADIUS * np.arcsin(np.sqrt(d))
return h # in the unit of km

def manhattan_distance(lat1, lng1, lat2, lng2):
a = haversine_array(lat1, lng1, lat1, lng2)
b = haversine_array(lat1, lng1, lat2, lng1)
return a + b

```

Listing 6.2: code2

6.2.3 plot.py

```
import seaborn as sns

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

def Hist(dataset, xlabel, filename):  #Log Trip Duration His
sns.set_style("darkgrid")
sns.distplot(dataset, kde=True, rug=False)
plt.legend()
plt.xlabel(xlabel)
plt.ylabel('Frequency')
plt.tight_layout()
plt.savefig('./fig/'+filename+'.png',dpi=100)

def CountPlot(dataset, column, label, s, a):   #Day of Week_Count_Plot
sns.set_style("darkgrid")
sns_plot = sns.factorplot(x=column,data=dataset,kind='count', palette="
    muted",size = s, aspect = a)
plt.xlabel(label)
plt.ylabel('Frequency')
plt.tight_layout()
sns_plot.savefig('./fig/'+label + '_Count_Plot.png',dpi=100)

def MapPlot(dataset, color = None, legend_plot = True):
```

```
26 sns.set_style("darkgrid")
27 city_long_border = (-74.03, -73.75)
28 city_lat_border = (40.63, 40.85)
29 if color is None:
30 sns_plot = sns.lmplot('pickup_longitude', 'pickup_latitude', data=
       dataset,fit_reg=False, scatter_kws={"s": 2.5}, legend =legend_plot)
31 else:
32 sns_plot = sns.lmplot('pickup_longitude', 'pickup_latitude', data=
       dataset,hue = color, fit_reg=False, scatter_kws={"s": 2.5}, legend =
       legend_plot)
33 #sns.plt.xlim(city_long_border)
34 #sns.plt.ylim(city_lat_border)
35 plt.xlabel('Longitude')
36 plt.ylabel('Latitude')
37 sns_plot.savefig('./fig/3_Map_Plot_PickUp.png',dpi=100)
38
39 def SpeedPlot(dataset,column,speed,label):
40 sns.set_style("darkgrid")
41 sns.pointplot(x=column, y=speed, data=dataset)
42 plt.xlabel(label)
43 plt.ylabel('Speed (km/hr)')
44 plt.savefig('./fig/4_Speed_By_'+label+'.png',dpi=100)
45
46 def PlotFeatureImp(feature_importance_table):
47 sns.set_style("darkgrid")
48 sns.factorplot(x="importance", y="feature_name",data=
       feature_importance_table, kind="bar")
```

```
49 plt.savefig('./fig/5_Feature_Imp_XGB.png',dpi=100)
50
51
52 def LonLatPlot(train):
53 sns.set(style="darkgrid", palette="muted")
54 f, axes = plt.subplots(2,2,figsize=(12, 12), sharex = False, sharey =
       False)#
55 sns.despine(left=True) # if true, remove the ax
56 sns.distplot(train['pickup_latitude'].values, color="m",bins = 100, ax=
       axes[0,0])
57 sns.distplot(train['pickup_longitude'].values, color="g",bins =100, ax=
       axes[0,1])
58 sns.distplot(train['dropoff_latitude'].values, color="m",bins =100, ax=
       axes[1,0])
59 sns.distplot(train['dropoff_longitude'].values, color="g",bins =100, ax=
       axes[1,1])
60 axes[0, 0].set_title('pickup_latitude')
61 axes[0, 1].set_title('pickup_longitude')
62 axes[1, 0].set_title('dropoff_latitude')
63 axes[1, 1].set_title('dropoff_longitude')
64 plt.setp(axes, yticks=[])
65 plt.tight_layout()
66 plt.savefig('./fig/6_Lat_Lon_Plot.png',dpi=100)
67
68 def ViolinPlot(train, y_, row_, hue_):
69 sns.set(style="darkgrid")
70 sns.violinplot(x=row_,
```

```
y=y_,
hue=hue_, data=train, split=True,
inner="quart")

def PivotPlot(train, y_, row_, col_):    #
    pickup_hour_and_pickup_weekday_Pivot
sns.set(style="darkgrid")
pivot_table=pd.pivot_table(train, index=row_, columns=col_, values=y_,
    aggfunc=np.mean)
sns.heatmap(pivot_table)
plt.tight_layout()
plt.savefig('./fig/'+ row_ + '_and_' + col_ + '_Pivot.png',dpi=100)

#Time_Distance_Plot
def Time_Distance_Plot(train):
sample_ind = np.random.permutation(len(train))[:5000]
sns.lmplot(x='distance_haversine', y='log_trip_duration', data = train.
    iloc[sample_ind], scatter_kws={"s": 10})
plt.savefig('./fig/8_Time_Distance_Pivot.png',dpi=100)

def Two_Hist_Plot(train, test):
sns.set_style("darkgrid")
sns.kdeplot(train['log_trip_duration'], shade=True, label = 'Train',
    color = 'b')
sns.kdeplot(test['log_trip_duration'], shade=True, label = 'Test', color
    = 'r')
plt.legend()
```

```
93 plt.xlabel('Log(Duration)')
94 plt.ylabel('Frequency')
95 plt.tight_layout()
96 plt.savefig('./fig/9_Two_His_Plot.png',dpi=100)
```

Listing 6.3: code3

References

[1] Traffic condition recognition using the k-means clustering method M. Montazeri-Gh, A. Fotouhi ∗ Systems Simulation and Control Laboratory, School of Mechanical Engineering, Iran University of Science and Technology, Tehran, P.O. Box 16846-13114, Iran Received 24 November 2010; revised 14 March 2011; accepted 7 June 2011

[2] url https://www.researchgate.net/publication/221019408

[3] url http://stackoverflow.com/

[4] url https://github.com/shiwang0211/NYC-Taxi