Fundamentals of Data Science (CS 834) Project Report

Project Title

"**Envisioning Yellow Taxi High Demand Areas in NYC city**"

Simranjeet Randhawa

ssr779@uregina.ca

Department of Computer Science

Professor

Alireza Manashty

Affiliation

University of Regina

Regina, Saskatchewan

Table of Contents

1. Team Member with a role:

Name: Simranjeet Randhawa

Group: Individual

Email Address: ssr779@uregina.ca

Expertise: Data Exploration, Data Visualizations using Plotty, Web Scrapping using Beautiful Soup and Modeling Data using Power BI or Data Studio, Designing Web Interfaces.

Detailed Role: I carried out the following tasks in my project:

- **Data Collection**: Data was collected from two different sources i.e TLC (Taxi and Limousine Commission) and Google Big query public dataset.
- **Data Storage** on Google Cloud Platform in parquet compressed form.
- **Data Access**: Made use of Big queries which are like SQL queries to access the data from google cloud storage.
- **Data Cleaning and Exploration:** Cleaned the rows that were outliers and carry out data exploration to discover insights from the taxi trips data.
- **Data modelling:** Joining together the weather data and trips data using nested joins in Big Query.
- **Feature Engineering:** Estimating which features are important to forecast the trips in NYC city. I used a correlation matrix to infer feature importance.
- **Model Selection and Evaluation:** Some models like Xgboost, Random forest, ARIMA, etc were selected and the accuracy of each model was calculated to select the best model for estimating the trips in different regions of New York for yellow Taxi.
- **Deployment of the Flask app:** Lastly, I was responsible for deploying the machine learning model on the cloud Google Cloud Platform.

2. Introduction:

New York City is known as the home of the great yellow taxis. Yellow taxi cabs are the main vehicles that reserve the privilege of street-hailing and prearranged passengers anyplace in NYC. The vehicles that come under TLC (Taxi and Limousine Commission) includes yellow taxis,

FHV's(For-hire vehicles), Green cabs, Paratransit Vehicle and Commuter Vans. Yellow cab is the most iconic as they are allowed to pick the passenger waving for a ride anywhere in the city. Whereas FHV provides only pre-arranged services and Green Taxi permits street hailing but is restricted to some zones only in NYC city. There are 13587 yellow cabs and each of them must have a medallion fixed to it. Taxi services, for example, Uber have upset the NYC taxi business. Uber has gotten gigantically famous in New York, and its excursions outpaced yellow taxicabs just because of a year ago.

The most concerning issue for any taxi company is to predict their tax's demand. Riders may need to stand by longer when request surpasses supply. There are currently there are around 13,500 yellow taxis in NYC. They may not satisfy all the interest yet it is conceivable to improve their usage. They may not fulfil all the demand but it is possible to improve their utilization. This past March, yellow cabs earned, on average, roughly $9,100 — a revenue drop of 36 per cent over six years. The fares per medallion dropped from $14,432 in March 2013 to $9,127 in March 2019 due to loss in the revenue generated by the yellow cabs taxis. [1] The figure below shows the Fares per medallion in NYC.

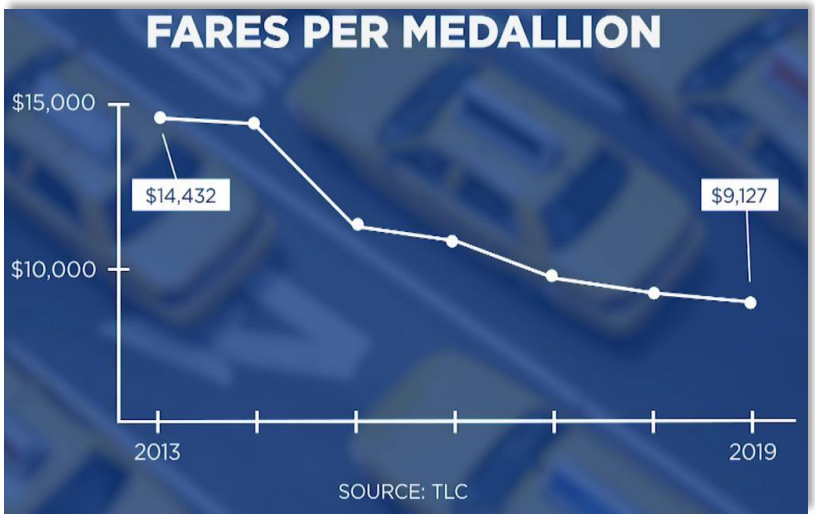


Figure 1: Yellow cab Fare per Medallion [1]

Yellow taxis pick up people in the street and FHVs(For-hire vehicles) pick up people through pre-arrangement. There is no pre-arrangement required in the case of Yellow Taxi indeed it allows street hailing pickups.

3. Business Problem Statement:

New York City is a very busy city and requires taxis quite often. I aim to predict the taxis pickups required in New York City depending on the weather conditions. The prediction can help the TLC agency to know the taxi demand in New York depending on the weather conditions and the NYC taxi company can prepare well ahead of time to make use of this prediction to ensure that there is the presence of certain taxis that are required in that weather. If the weather is extreme there is no point dispatching a large number of taxis as it will only cause loss to the company revenue. And if the weather is pleasant there is a need for a certain number of taxis such that the demand of the customer is met. If the demand is not met the company is certainly to lag behind Uber and other taxi services.

Figure 2: Line Chart showing Departed Trips in New York City [2]

Till 2014 the trips of yellow taxis were growing but in 2015 with the introduction of uber. the yellow taxi daily dispatched trips began to decrease gradually. A unique quality about Yellow cabs is that they can pick up passengers from the streets without any pre-arranged booking required. Whereas in the case of Uber, passengers mainly use the app to arrange for the taxi. So I found an opportunity for yellow cabs to get back on track by targeting the streets where the pickups demand are high. If the passenger waits too much they switch to apps to look for a taxi but if the passengers can see a yellow cab right in front of them, they will surely choose the yellow cab instead of waiting for Uber. Hence this data science projects focus to help the NYC TLC yellow taxis to get

back on track and increase the number of pickups around the NYC city. Also, the yellow cab taxi drivers can make use of the website to see the forecasted demand for different zones in NYC city.

4. Solution Overview:

I have implemented my project in the following way:

Step1: I discovered the data from the TLC taxi trip website and then I retrieved the data from the AWS S3 bucket and stored all the data on Google Cloud Platform using Spark and Pandas.

Step 2: I made Use of Big Query to analyze the data set and get only relevant rows from the dataset. I also made use of Big Query and pandas data frame to clean the data as well.

Step 3: Carry out data exploration to find out the most demanding areas in NYC City and also to know how weather affects the count of NYC yellow taxi trips in NYC City.

Step 4: Did Feature Engineering to get an effective outcome. Use methods like One hot encoding and binning in feature engineering.

Step 5: Divided my data into train / test and also used cross-validation to get the optimal result from the model. The model used are ARIMA, Xgboot, Random Forest Regression, Linear Regression, Light GBM, etc. I used the Grid Search CV to get the best result from all the models shown above.

Step 6: Published a fully functional website using Flask that enables the yellow cab taxi drivers and TLC business to leverage the website to enhance their profit.

Step 7: Prepared Dashboard that can help the business know the weekly pickups in advance.

Components of the project:

The components of the projects are as follows:
Data Strategy: It deals with the data collection: what data is collected and why. So my data strategy was as follows:

- **What Data:** I needed the taxi trips record for a yellow taxi cab in New York city to instantiate my work. The taxi trips data for the last two years are sufficient to forecast the taxi demand in the upcoming year. So I was motivated to choose a dataset of yellow cab

taxi trips in NYC city in the last two years i.e. 2018 and 2019. This dataset was open data and was publicly available on the NYC TLC website as a public dataset.

- Why this Data: My business goal is to improve the business of TLC yellow cabs which at present is undergoing loss because their drivers are not finding many passengers as they are not standing in the right areas of NYC city. Some area has too much demand but has fewer taxis available there and some areas have less demand but that area has a large number of taxis available. In some areas, demand is more but demand is also dependent on which hour it is. For example, If a taxi driver goes to a high demand zone area at noon, it might be possible that the drivers don't find any pickups at that time. Hence this data will help to forecast the taxi picks required in different zones considering all day, the hours and weather.

Data Engineering: The data consist of more than 50 million rows and weather data is on Google public data set and taxi 2019 data is taken from TLC open dataset and 2018 taxi data again from Google public dataset. Hence this is an essential component in my project to access, organize and use the data. To make the data engineering very easy, I decided to make use of the Google cloud platform. To make access to the data easy, I upload the large data set to the Google Cloud Big table so that I can easily access it using Big Query on Google Cloud. I organized the entire data in one database and used it in an iPython notebook using the support of Big query. Hence this component was easily handled making use of Google Cloud Platform.

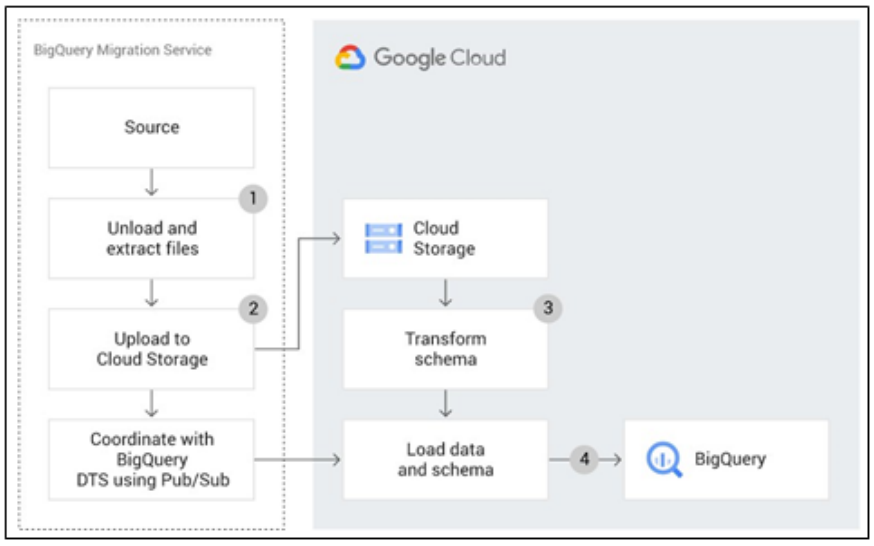


Figure 3: High-Level Diagram showing BigQuery Functionality

BigQuery is SaaS(Software as a Service) that enables analysis of huge datasets working in parallel with Google Cloud Storage. It can also be used complementarily with Map Reduce.

Data Analysis and Model:

As the data access part has been covered in the Data Engineering components now the focus shifts to getting insights from the data and then prepare a model to forecast the demanding areas around NYC city. For getting insights, I made use of three things: Big query to get month wise records to explore. Data Studio and Python plotty library to explore the data to gain any valuable insights and lastly data studio to visualize the entire data set as Power BI is not able to support such a large data set. For modelling, I collected the clustering information of the NYC city into regions having a high number of taxis pickups requirements and then make use of Xgboost regressor to predict the number of pickups required in each cluster formed using K means algorithm.

I also performed feature importance and feature selection using Xgboost so that I could identify which feature is important to consider to predict the number of picks. For example, the hour of the day will certainly be an important feature to consider. But this became more clear once I perform the feature selection step. I did not focus on just one algorithm, I used certain alternative like the random forest. Apart from regression I also used ARIMA to forecast the taxi trips around the city based on the previous two years data. Model selection is the last phase in which I selected the most suitable model based on the accuracy score.

Data Visualization and Operationalize:

My main goal was to create a fully published and developed website that will incorporate all my results into one place. I made use of Flask to create a web app and deploy it on Google cloud because of its cheap cost depending.

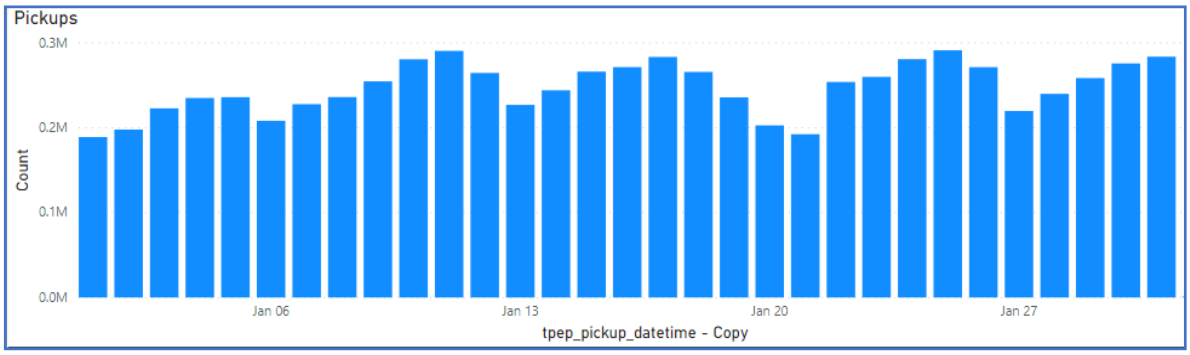


Figure 4: Jan 2019 Pickups vs Pickup Date Bar chart (Power BI)

For data visualization, I made a dashboard that will help the TLC agency to look at the records with ease. A sample of visual is given below which illustrate the count of the pickups in entire NYC city in from Jan 01 to Jan 31 in 2019.

Entire Workflow High-Level Diagram

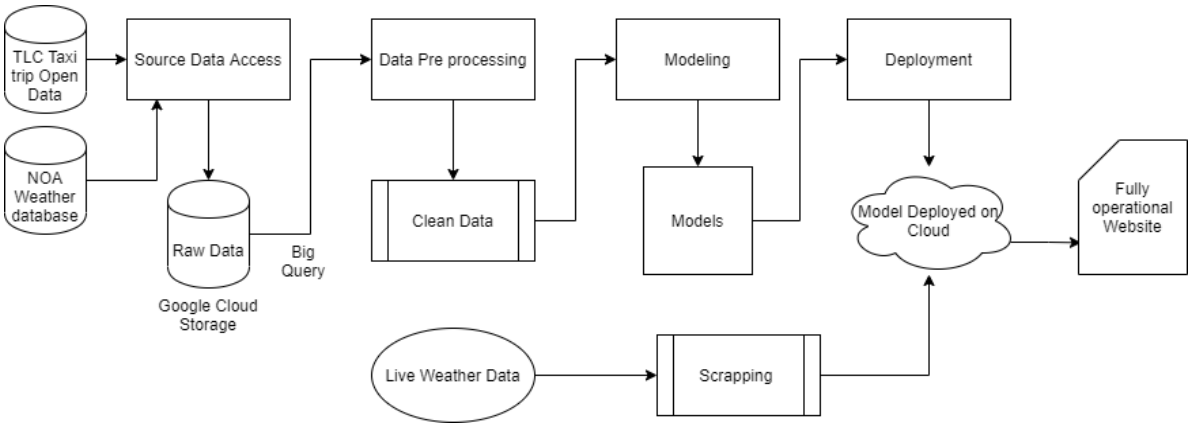


Figure 5: High-Level Diagram reflecting Solution Overview

Expected Outcomes: A fully developed and operational cloud-based website/app showing a geographical map displaying all the high demand areas in NYC city based on the previous year's data. The figure below shows the zone in New York City. The fully developed website will display the zones with high taxi demand requirements. Also, the live weather of the zone will be scrapped from Google and will be taken into account for predicting the number of taxis pickups estimated around New York City.

Figure 6: Zones in New York City

5. Data (Source, Format, Access, Statistics, and Storage)

Source:

NYC Taxi Trips Data Source:

NYC Government provides TLC trip record data and provides the links to download taxi trips data for each month. The links are provided as follows:

2018 Yellow Taxi Data Set:

Month 1: Link is: M12018

Month 2: M22018 and so on.

Link: https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_yyyy-mm.csv

(where YYYY-year, MM – month). Similarly, the 2019 dataset is also provided by TLC.

Weather Data: The Global Surface provides the summarized version of weather dataset. This dataset is available publicly. It was created by NOAA for the purpose of keeping the weather history intact. It includes all the weather data from the year 1929. It is updated daily and is collected from over nine thousand stations.I will be targeting two tables in this database one ids gsod2019, gsod2018.

Zone Data: The TLC yellow taxi zone data is also provided by the NYC government on Taxi zone. It is an open data set that is publicly available.

Format:

1. TLC Yellow Cab Data is a dataset that is structured and available in CSV format.
2. New York Weather Data is a dataset that is structured and available in CSV Format.
3. Taxi Zone Operating data is also in CSV format.

How I Accessed the Data Initially:

The data is stored on Amazon s3 instance which makes it easy to retrieve. I may use pandas data frame command to retrieve the data but the data was very large, therefore I retrieve it month by month in a pandas data frame and then combine and stored it to my google drive in the form of parquet file.

```
import pandas as pd

df1=pd.read_csv("https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2019-01.csv")
```

I also made a copy in Google Cloud Data Warehouse because Big query helps to access the data from Google cloud at high speed. This notebook will illustrate the process of accessing the data from the Google cloud:

https://github.com/ssrbazpur/Data-Science/blob/master/Access_the_Data.ipynb

Statistics:

All data is stored in the Google Cloud. The statistics of all the data tables that I used in my project is as follows:

TLC Taxi trips Data:

2018:

Table ID nyc-taxi-265120: NYC.2018firsthalf
Table size 6.58 GB
Number of rows 53,925,735
Number of Columns:17
Table expiration: Never
Last modified: Mar 5, 2020, 7:43:41 PM
Data location: US

Table ID: nyc-taxi-265120:NYC.2018SecondHalf
Table size: 7.15 GB
Number of rows: 48,878,515
Number of Columns:17
Table expiration: Never
Last modified: Mar 5, 2020, 9:48:06 PM
Data location: US

2019:

Table ID: nyc-taxi-265120:NYC.2019firsthalf

Table size: 6.8 GB

Number of rows: 44,459,136

Created: Mar 5, 2020, 7:44:44 PM

Table expiration: Never

Last modified: Mar 5, 2020, 7:44:44 PM

Data location: US

Table ID: nyc-taxi-265120: NYC.2019secondhalf

Table size: 6.13 GB

Number of rows: 39,939,883

Number of Columns:17

Table expiration: Never

Last modified

Mar 5, 2020, 9:01:56 PM

Data location: US

Weather Data

2019:

Table ID: bigquery-public-data:noaa_gsod.gsod2019

Table size: 734.3 MB

Number of rows: 4,120,820

Created: Jan 11, 2019, 11:03:04 AM

Table expiration: Never

Last modified: Jan 30, 2020, 11:46:05 AM

Data location: US

2018:

Table ID: bigquery-public-data:noaa_gsod.gsod2018

Table size: 723.88 MB

Number of rows: 4,010,814

Created: Jan 10, 2018, 1:49:27 PM

Table expiration: Never

Data location: US

Taxi Operating Zones Data

Table ID: bigquery-public-data.new_york_taxi_trips.taxi_zone_geom

Table size: 2.49 MB

Long-term storage size: 2.49 MB

Number of rows: 263

Created: Nov 30, 2018, 3:22:34 PM

Table expiration: Never

Last modified: Nov 30, 2018, 3:24:38 PM

Data location: US

Storage:

Figure 7: Google Storage

Google Cloud Storage offers 12 years of free services with 400 $ credit. Hence I have stored my entire data on Google Cloud. I have stored some month's data on Google Cloud in the form of parquet files as well to keep a backup.

6. Data Analytics Lifecycle:

This project covers all the steps from the Lifecycle of Data Analytic which is depicted as follow:

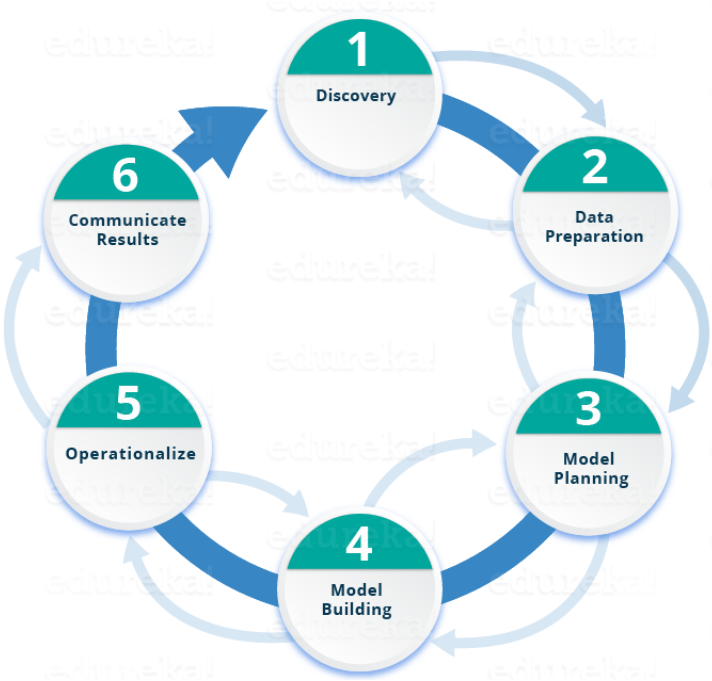


Figure 8: Data Analytics Lifecycle

6.1 Data Discovery:

Here, I am assessing that if I have the required resources to sustenance the project. In this phase, I also need to formulate the initial hypotheses (IH) and I have to frame the business problem to test. The key activities in the Data Discovery Phase that I made sure of:

1. Drafting the business problem statement.

A unique quality about Yellow cabs is that they can pick up passengers from the streets without any pre-arranged booking required. Whereas in the case of Uber, passengers mainly use the app to arrange for the taxi. So I found a business opportunity for yellow cabs to get back on track by targeting the streets where the pickups demand are high. If the passenger waits too much they switch to apps to look for a taxi but if the passengers can see a yellow cab right in front of them, they will surely choose the yellow cab instead of waiting for Uber. Hence this data science projects focus to help the NYC TLC yellow taxis to get back on track and increase the number of pickups

around the NYC city. Also, the yellow cab taxi drivers can make use of the website to see the forecasted demand for different zones in NYC city.

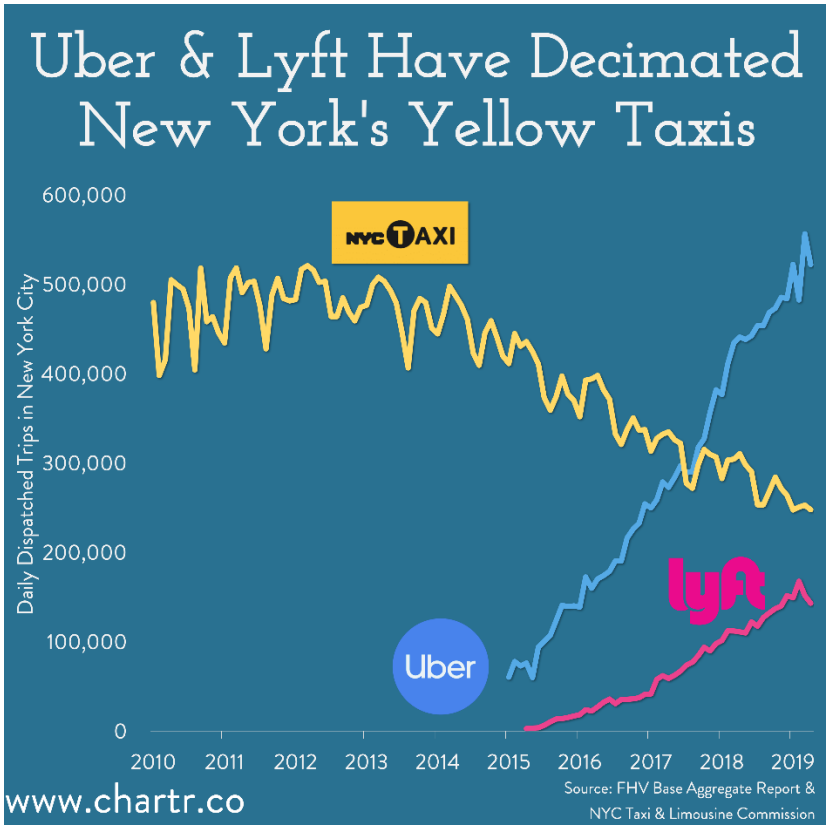


Figure 9: Line Chart showing Departed Trips in New York City [2]

2. Assess resource needs and availability:

Resources Needed: Yellow Taxi pickups of past years and weather data of past year day by day was needed to make any projection in the future. Also, I need the taxi operating zone dataset of NYC city.

Resources Availability: Yellow taxi dataset is a publicly available dataset. The attached datasets were provided and collected by the TLC in order to keep the records of each trip in New York city for the yellow taxi cabs.

Current Situation:

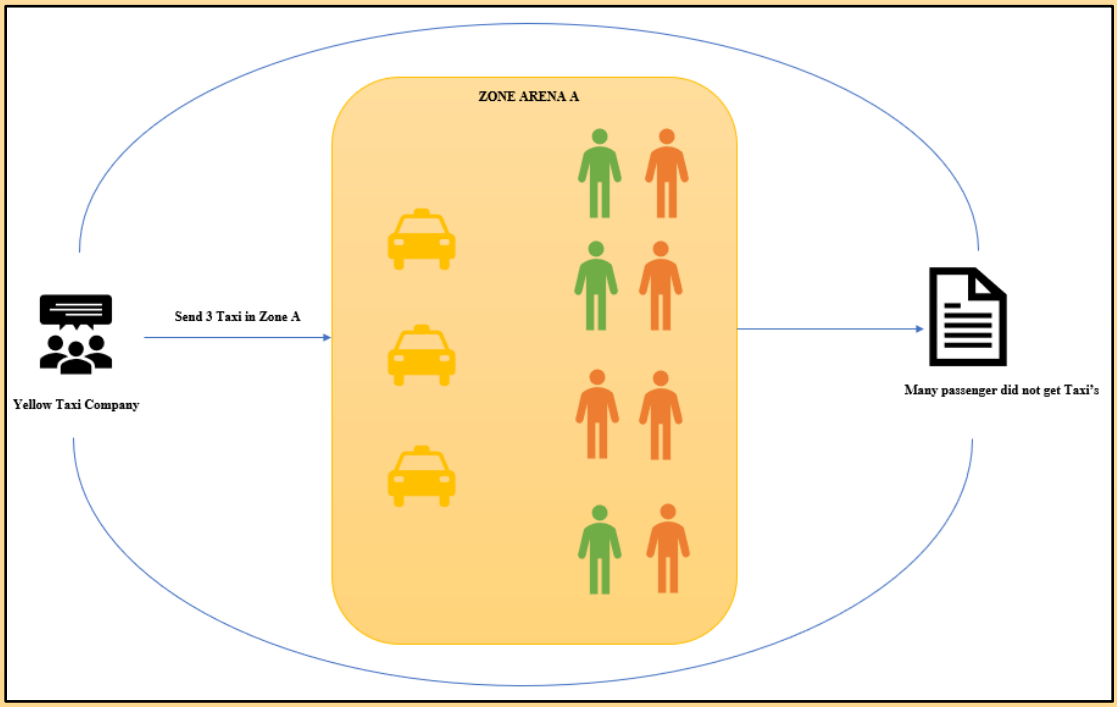


Figure 10: Passenger Demand more but fewer taxis

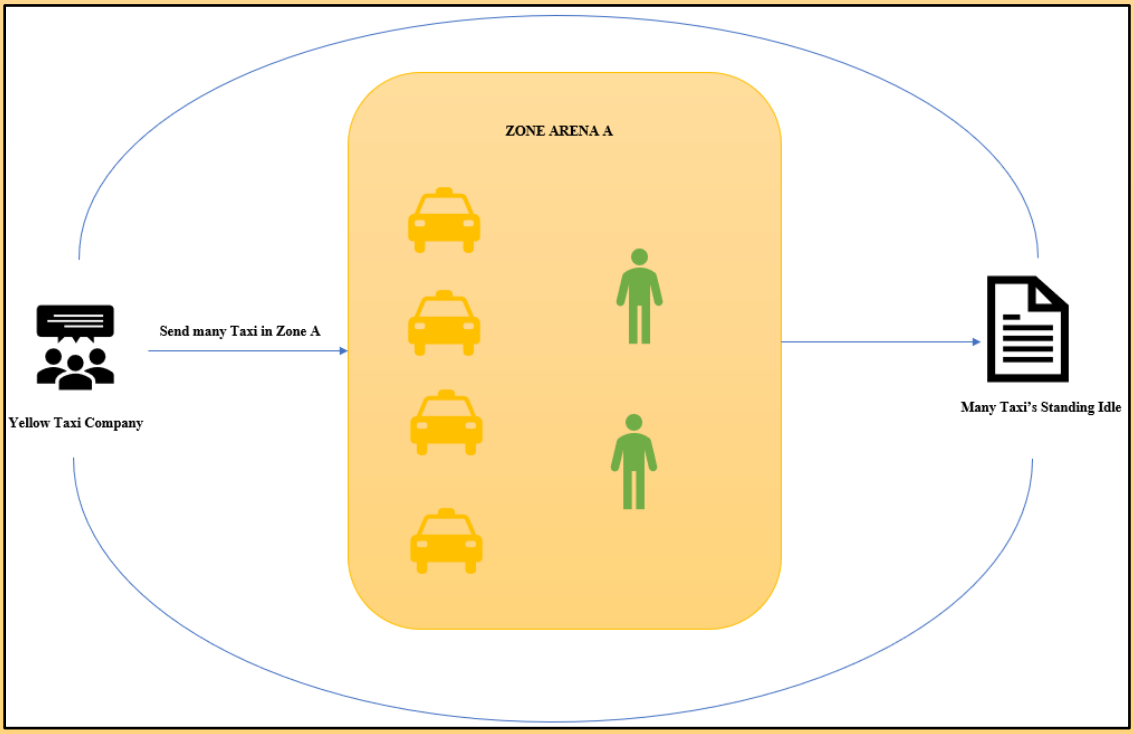


Figure 11: Many Taxies were standing idle as passenger demand was less

Desired Situation: (The taxi company knows beforehand how many taxies to send)

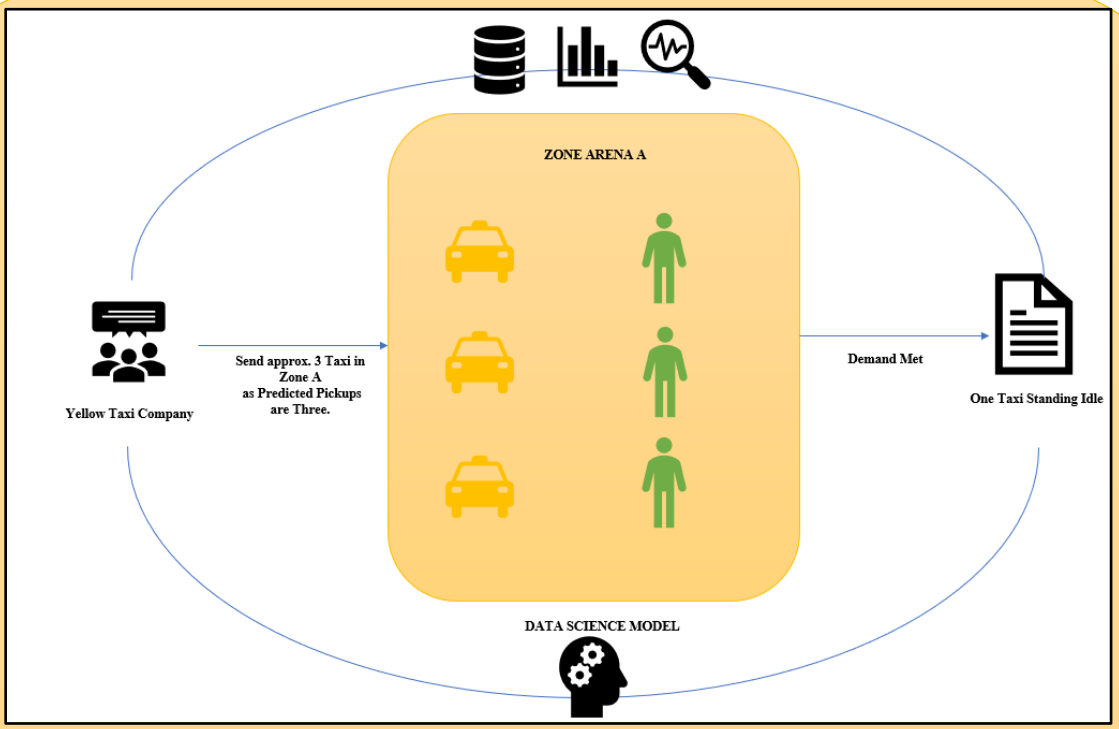


Figure 12: Meeting Taxi Demand when fewer passengers

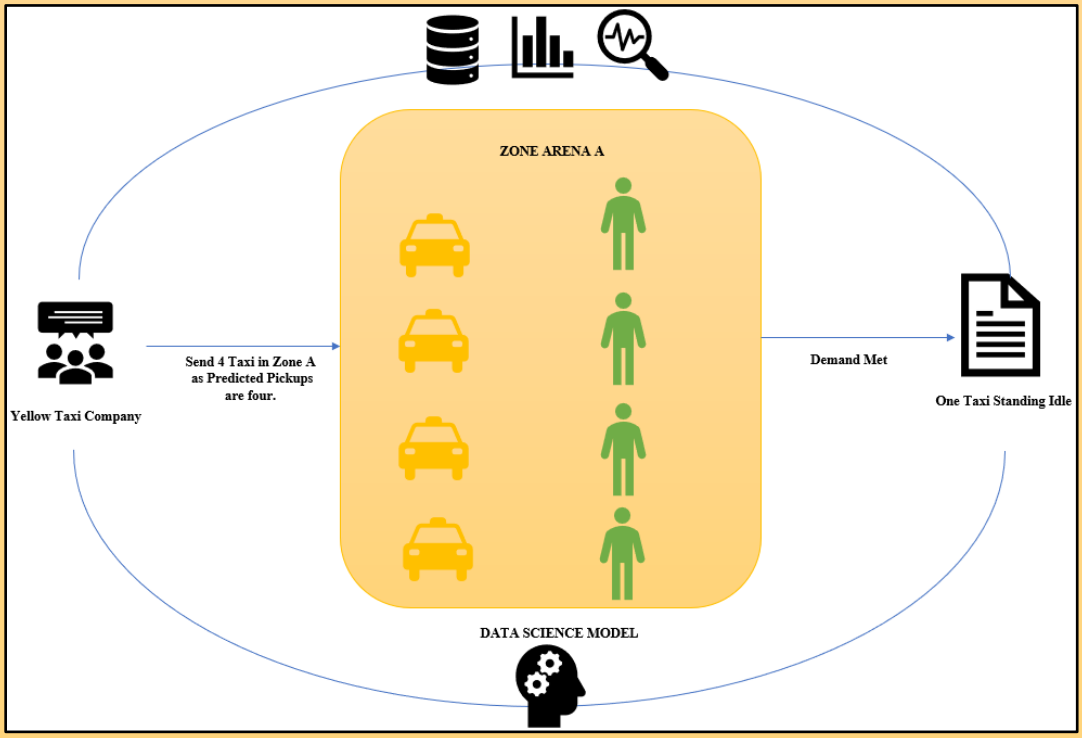


Figure 13: Meeting Taxi demand for more passenger

3. Reframe the problem as an analytics challenge.

The problem demands to predict the pickups for different taxi operating zones in the NYC city. Also, the data demand a lot of cleaning and exploration as the dataset is huge and cannot be trained without carrying out cleaning and exploration. There is also another challenge to store and analyze such a big dataset. Though it is not a big data problem, it gives the taste of it.

4. Drafting an analytics plan:

The analytics plan for my project is as follows:

1. Data Gathering
2. Data Cleaning and Exploration
3. After that Model Planning
4. After planning go to Model Building
5. Operationalize by building a Flask application.
6. Communicate the results through the dashboard and website to the stakeholders.

Data Gathering:

Gathering of data is an important step in the life cycle of machine learning. The goal of this step is to identify and obtain all data-related problems. In this step, I identified the different data sources, as data can be collected from various sources such as files, databases, internet, or mobile devices. This one step is the most important in the life cycle of a data science project. The feature and amount of the collected data will define the efficiency of the output. The more will be the data, the more accurate will be the prediction. This step includes the below tasks:

1. Identify various data sources
2. Collect data
3. Integrate the data obtained from different sources

The data is obtained from various data sources and collected data using Pyspark. Lastly, I integrated data obtained from different sources. By performing the above task, we get a coherent set of data, also called a dataset. It will be used in further steps. I also stored this data in the parquet files. These parquet files were compressed so it enables the ability to be loaded in pandas dataframe. As in the previous instance, I was getting Not enough memory error.

Learning about Data Sources

Weather Data: The Global Surface provides the summarized version of weather dataset. This dataset is available publicly. It was created by NOAA for the purpose of keeping the weather history intact. It includes all the weather data from the year 1929. It is updated daily and is collected from over nine thousand stations.I will be targeting two tables in this database one ids gsod2019, gsod2018.

Zone Data: It consists of a Shapefile that is converted into a CSV file using an online converter and then processed using pandas. It consists of 263 zones in NYC city where the yellow taxi operates.

Yellow Taxi Trip Data: NYC Government provides TLC trip record data and provides the links to download taxi trips data for each month. The TLC has made the taxi trips data in New York city public with the aim to keep track of all the taxi trips done over the time. This dataset all contains some erroneous entries which is to be explored and removed from the dataset.

Data Collection so that Data preparation can be carried out on the Data:

The dataset of yellow cabs includes many columns. Some of the columns signifies the total amount paid by the passenger for the trip, the pickup location of the passenger, the drop off location, the time of the trip, etc. It is available as an s3 amazon data bucket which was initially difficult to read using a pandas data frame. Hence the requirement of Pyspark was felt that eased the task of collecting the data from the s3 bucket and use the concept of RDD to help process a large set of data.

It was giving Not enough memory error. I decided to go with Pyspark which uses the RDD architecture which consumes less RAM. Also, I had set up the environment for RDD and also upgraded my RAM TO 25 GB with a standalone cluster on Google Colab.

Working of Spark ?

The main functionality is that it is based on lazy evaluation which means the computation only happens when some action is performed on the dataframe. This helps to make use of minimum RAM and give the optimal result. Also each task is executed on dedicated clusters such that parallel computing can happen. It also keep track of key task of scheduling.

It also offers support of structured SQL which makes the exploration of dataset very easy. It uses RDD architecture in the backend to process the dataset and also support real time data analysis unlike Hadoop. It is more reliable and faster than Hadoop distribute sytem. Therefore I decided to make use of PySpark.

Data Collection using Pyspark (This was done because the pandas' Data frame was running out of memory):

After installing PySpark it is essential to create a Spark Session where the work is to be done. The following creates a session:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.master("local[*]").getOrCreate()
```

After creating the spark context I added all the monthly data to the spark Context:

```
from pyspark import SparkFiles

url1 = "https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2018-01.csv"

spark.sparkContext.addFile(url1)
```

Reading CSV using Pyspark using the mechanism of RDD:

```
df1 = spark.read.csv("file://"+SparkFiles.get("yellow_tripdata_2018-01.csv"), header=True, inferSchema= True)
```

In this way, I read all the datasets and then performed union operation using Spark. Once all the dataset of 2018 and 2019 Taxi trip was collected in the Spark data frame then I stored in into a compressed Parquet file format that helps me access the file at ease. I uploaded the entire data that I collected using PySpark in the Google Big Table that provided me with Big Query which helps to query massive big dataset in a matter of seconds.

Using the Following I transferred data in parquet files to the Google Cloud Big Table:

```
#Storing to Google Cloud Platform

project_id = 'nyc-taxi-265120' #@param{type:"string"}

y2019upper.to_gbq('NYC.2019secondhalf',
```

```
        Context.default().project_id,
        chunksize=500000,
        if_exists='append',
        verbose=False
        )
```

The screenshot of the shape of the Taxi trips dataset are as follows:

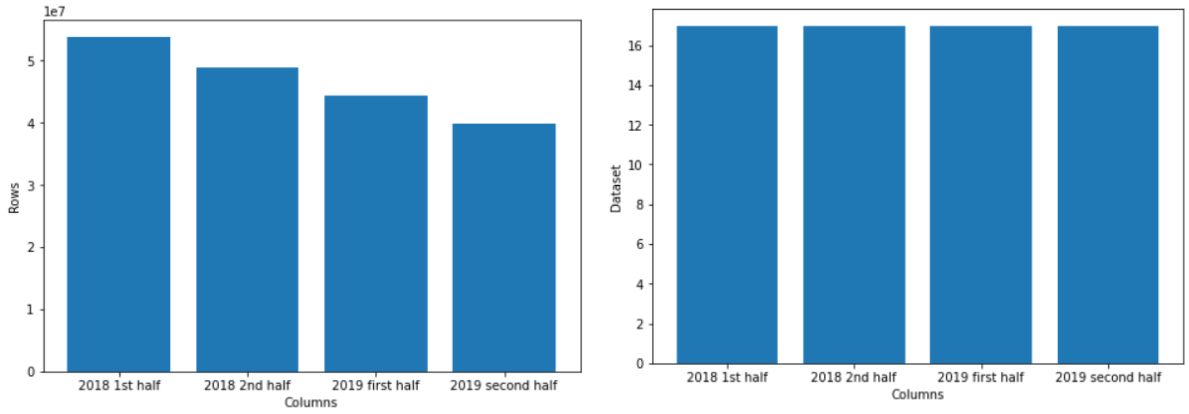


Figure 14: Rows and columns in Taxi Trips Dataset

Table Information for all The Data Stored on Google Cloud:

Table info

Table ID	nyc-taxi-265120:NYC.2018firsthalf
Table size	6.58 GB
Number of rows	53,925,735
Created	Mar 5, 2020, 7:43:41 PM
Table expiration	Never
Last modified	Mar 5, 2020, 7:43:41 PM
Data location	US

Table info

Table ID	nyc-taxi-265120:NYC.2018SecondHalf
Table size	7.15 GB
Number of rows	48,878,515
Created	Mar 5, 2020, 8:45:31 PM
Table expiration	Never
Last modified	Mar 5, 2020, 9:48:06 PM
Data location	US

Figure 15: 2018 First and Second Half Table Statistics

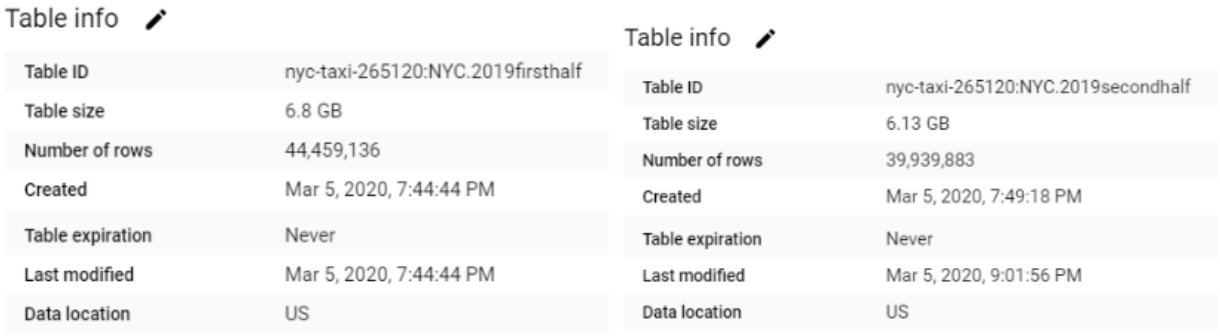


Figure 16: 2019 First and Second Half Table Information

Table & Schema Information for Zones in NYC city

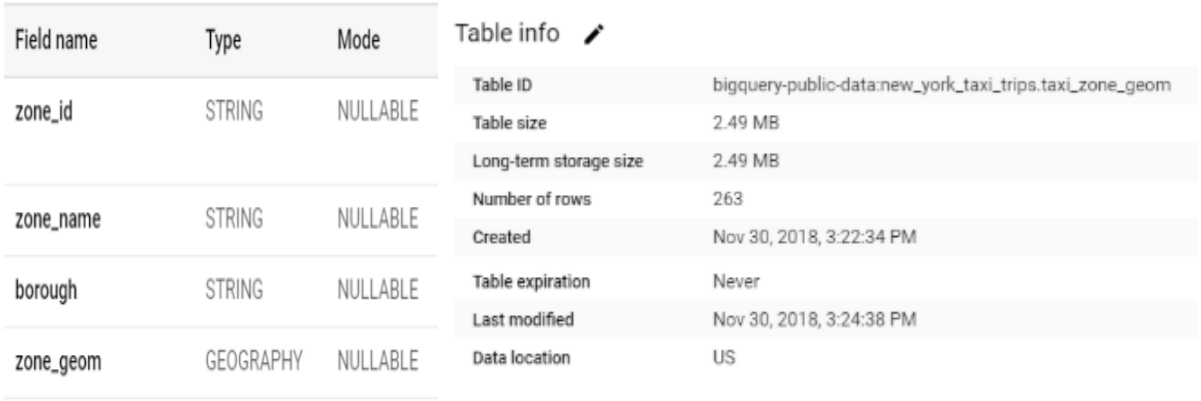


Figure 17: NYC city taxi zone Table Information

Table Information for Weather Data

Figure 18: Table Information for Weather Dataset

Understanding Discovery: Revelation is a data-gathering process intended to dive profound into the details of what is critical to a customer's business, target audience, and industry. The extension

and profundity of research and request will contrast from task to extend, yet the outcomes are the equivalent: important information. The more data you assemble, interpret and comprehend, the more set you up will be to execute a site on budget plan and target.

6.2 The Data Preparation Phase:

During this phase, I prepared an analytical sandbox which will give me the ability to perform ETLT. I decided to use the Big Query Analytical Sandbox which provides ETLT support. Establishing the analytical sandbox and then performing ETLT and then data conditioning and finally surveying and visualization to see that the required data is in an appropriate format.

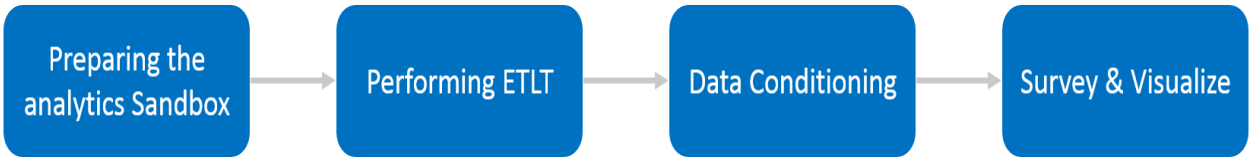


Figure 19: Data Preparation Phases

Establishing an analytic sandbox:

As I transferred all my data to Google Cloud. I made use of the Google Cloud platform that provides with the BigQuery sandbox. The BigQuery sandbox gives you free access to the power of BigQuery subject to the sandbox's limits. The sandbox allows you to use the web UI in the Cloud Console without providing a credit card. You can use the sandbox without creating a billing account or enabling billing for your project. Big Query provides the ability to deal with big data using SQL like queries. In Hadoop, I used to use Hive to get back the result. So I decided to use Big Query in this project which gives the ability of ETL.

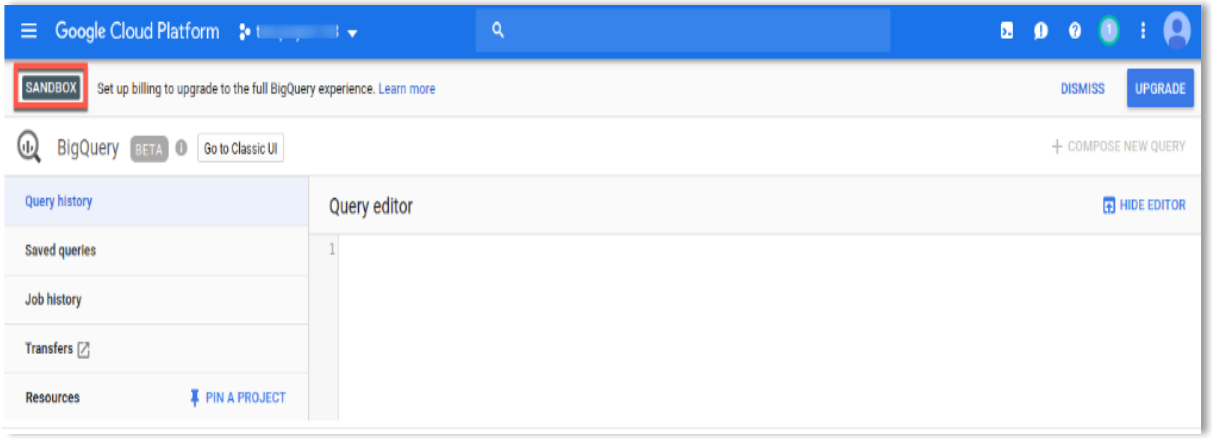


Figure 20: Building an Analytical Sandbox

Extract, Transform, Load, and Transform (ETLT):

Extract : We can extract the data using simple query. The below retrieves the all the entire table and then store into a pandas data frame for transformation. Extracting 2018 Jan-June Dataset:

```
query = """
SELECT * FROM `nyc-taxi-265120.NYC.2018firsthalf`
"""

df = client.query(query).to_dataframe()
```

Transforming the Data frame: The below query helps to transform the data. Like for instance the below query takes first half of 2018 and removes rows where total_amount column is less than 0 as no passenger can give less than 0 dollar to the cab driver. The query is also joining the taxi trip dataset with taxi zones.

```
query = """
with td as(
WITH AB AS
(
SELECT  CAST(tpep_pickup_datetime AS DATETIME) AS PICKUP,* FROM `nyc-taxi-265120.NYC.2018firsthalf` where total_amount>0
)
SELECT  PICKUP,CAST(EXTRACT (YEAR from PICKUP) AS STRING) AS year,EXTRACT (DAYOFYEAR from PICKUP) AS daynumber,EXTRACT (HOUR from PICKUP) AS hour, cast(PULocationID as STRING) as LOCATION,COUNT(*) AS numtrips FROM AB group by PICKUP,year,daynumber,hour,LOCATION),points AS
(SELECT *,ST_CENTROID(zone_geom) as p FROM `bigquery-public-data.new_york_taxi_trips.taxi_zone_geom`)
SELECT cast(DATE(td.PICKUP) as STRING) AS A1,
```

```
points.zone_id,CAST(TIME(DATETIME_TRUNC(td.PICKUP, HOUR)) as string) rounded_to_
hour,
SUM(td.numtrips) as label,
FROM  td
INNER JOIN
points ON points.zone_id=td.LOCATION AND
eXTRACT (YEAR from PICKUP) =2018
group by zone_id,A1,rounded_to_hour
"""
df1 = client.query(query).to_dataframe()
```

As show above ,BigQuery gives the ability to transform the dataset.

Load: After transformation, the data can be loaded to the Big query data warehouse using the following:

```
b1.to_gbq('hello.2018DataupdatedFirsthalf',
          Context.default().project_id,
          chunksize=500000,
          if_exists='append',
          verbose=False
          )
```

Transform: The above dataset can be again transformed and the column data types can be easily changed using BigQuery. Hence Big Query gives the ability of ETLT. This was about the analytics sandbox that I used for this project. Now coming to other sections in Data Preparation.

Data preparation is termed as the process of cleaning the data so that the data can be transformed into a form that can be processed and analyzed. This step is essential in any Data Science project as without it it is not possible to build an efficient model. The raw data was thoroughly visualized and explored to detect any outliers. Big query made it easy to explore the big dataset. However, just detecting outlier and cleaning the data does not solve the purpose. It is essential to carry out data modelling and wrangling in order to make the raw data into an appropriate format so that it can be trained or feature engineering can be done on the variable prior to the model building phase. Thus this phase is essential.

Visualize before Analysing:

The visualization of the zone dataset to check that the zone is within the boundaries of NYC city.

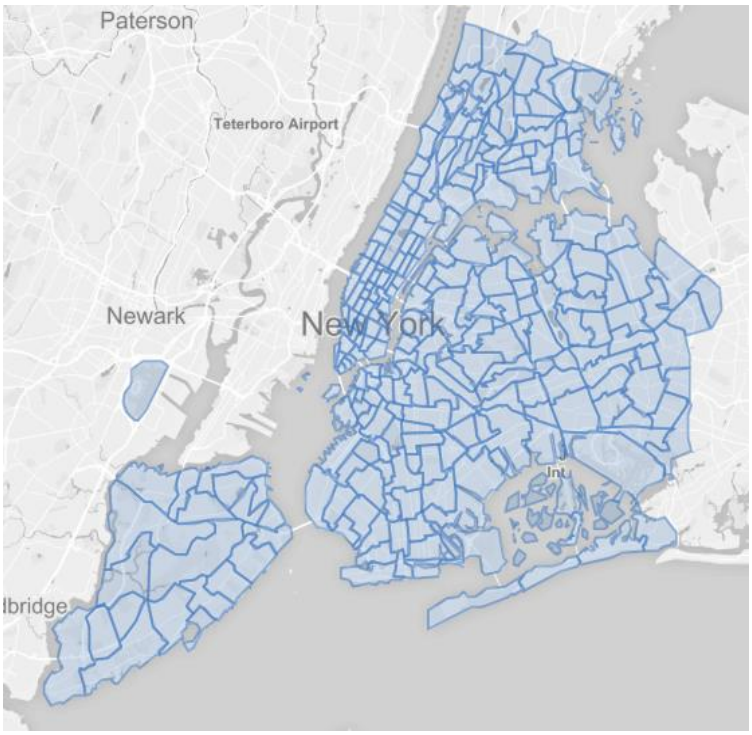


Figure 21: Zone Data Set Visualization

The geojson file format could be easily visualized using Power BI and even Github can read the Geojson file. So after checking and getting to know that the zones are correctly plotted, I decided to move to the taxi trips dataset.

The histogram shown below shows the amount paid by the passenger and from the visualization, it can be detected there are few negative amounts paid by the passengers. (Please note this visualization was carried only to look for outliers. Therefore I have considered the only sample from the complete dataset.)

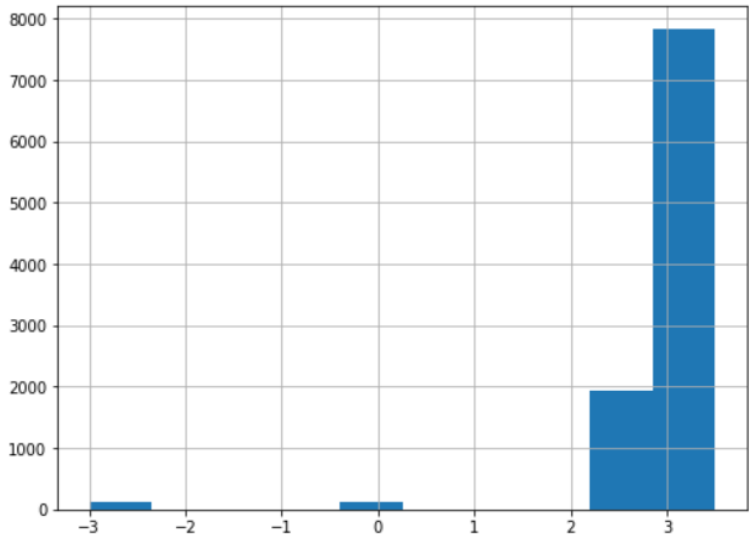


Figure 22: Histogram Plot showing there are negative amount given by the passenger

I also analyzed the relationship between two variables using a Density plot. From the plot, it can be seen which zone id has got the maximum amount from the passenger. This plot was constructed only by the sample of data. For considering the full dataset I used Big Query for analyzing the trend.

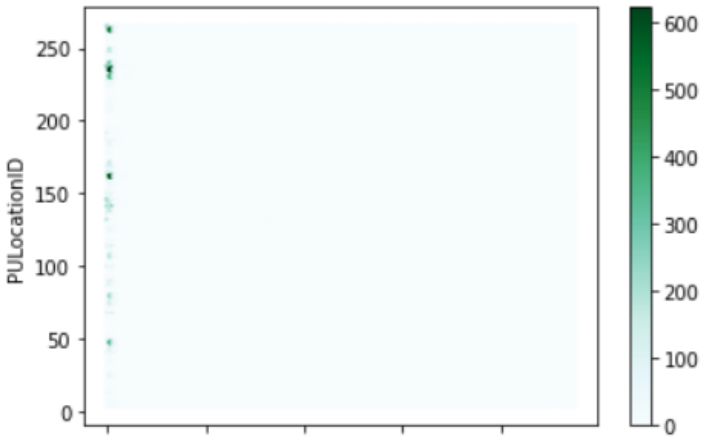


Figure 23: Hexabin Plot to show the demand pickup Locations

I also used a density plot and log of density plot to look at how much a passenger pays and the payment amount is distributed.

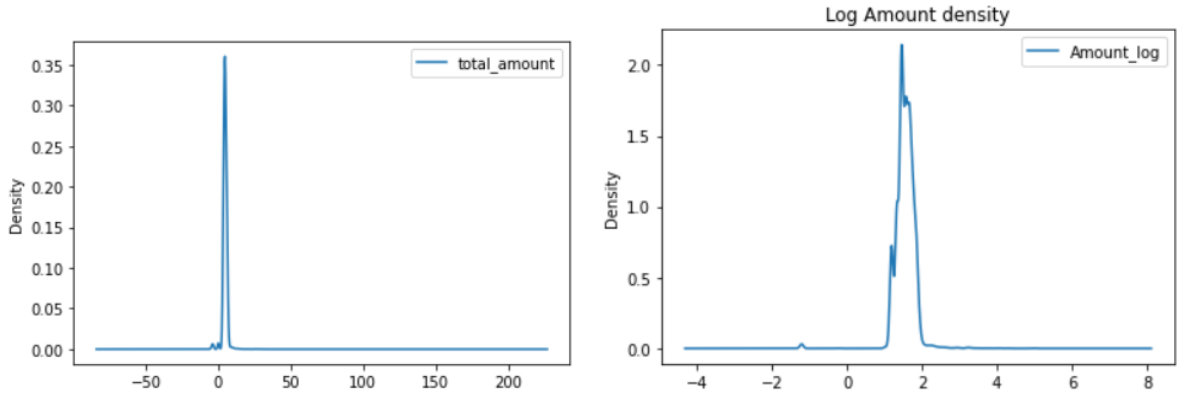


Figure 24: Distribution Plot and Log Distribution of Amount

I used the histogram plot for every column (NYC second half sample dataset). The histogram was used to study the distribution of a single variable and also to look for outliers.

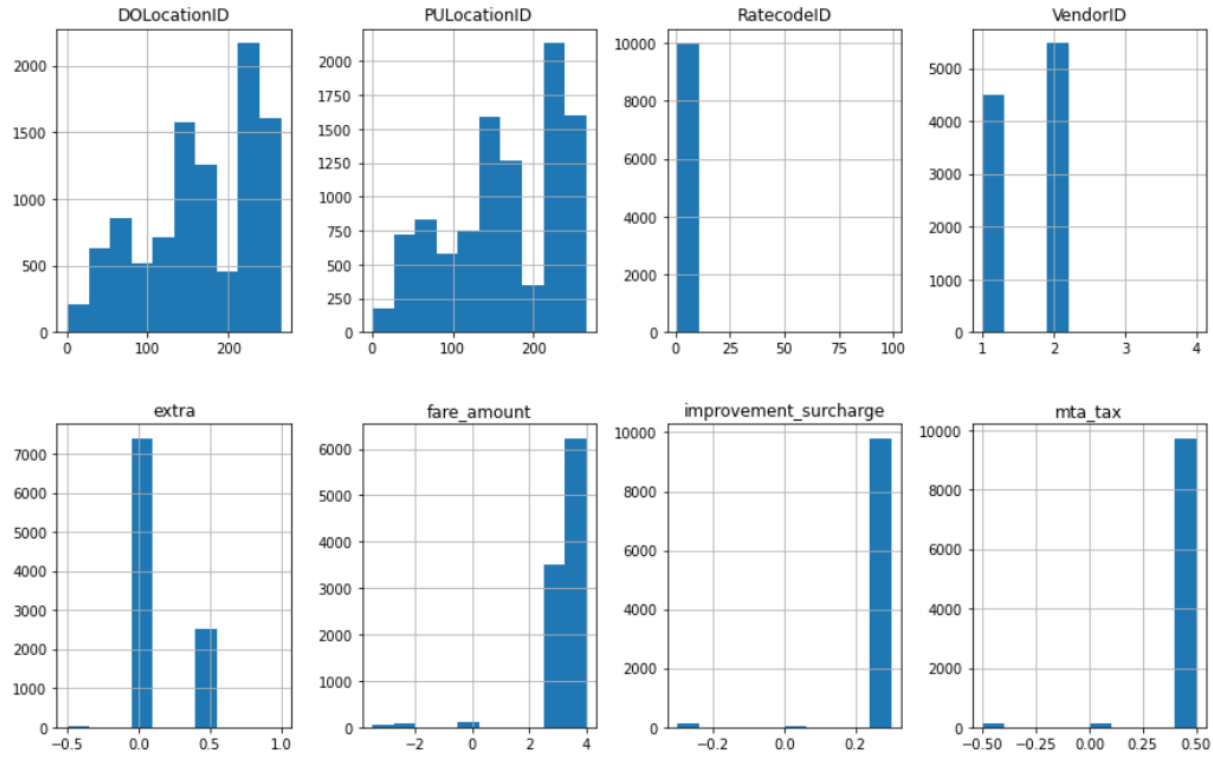


Figure 25: Histogram Plot to study the distribution of a Variable

These plots gave me the ability to see how the distribution of a single variable. For example, the pickup and drop off zone ID are the same in most cases. The passenger that gets in one zone is expected to be dropped in the same zone ID in most cases. But thorough exploration is also required before reaching to a conclusion. Hence more visualizations are required to see the relationship.

Joint Distribution Plot

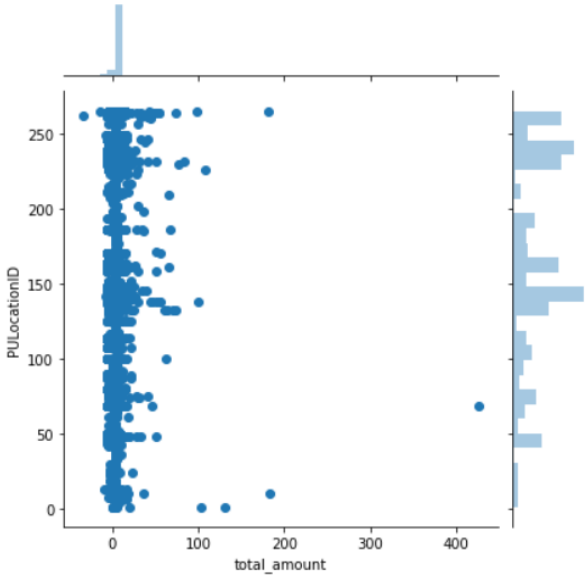


Figure 26: Joint Distribution Plot between Passenger Location ID and the total amount

I studied the joint distribution plot that servers two purposes. It also shows the scatter plot and the distribution plot on the sides. Hence it gives the ability of two plots in one to study the distribution and relationship.

Data Cleaning:

2018 First Half Yellow Cab Data Cleaning (Big Query)

1. Analyze if there are rows with total trip amount < 0?

```
query = """ SELECT * FROM `nyc-taxi-265120.NYC.2018firsthalf`
                where total_amount < 0 """
df_head = client.query(query).to_dataframe()
```

This query returned Number of rows with amount of trip <0: 29860. So in total there are 29860 rows where amount given for the trip is less than zero.Below is a scatter plot that shows there are negative total_amount in many Pickup location zones.

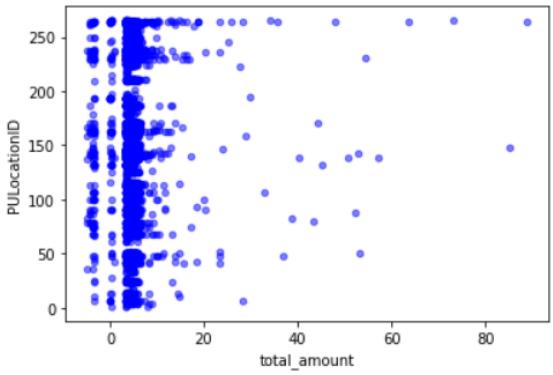


Figure 27: Scatter Plot Visual

2. Are there rows with pick up time > drop off time?

```
query = """
SELECT * FROM `nyc-taxi-265120.NYC.2018firsthalf` where tpep_pickup_datetime > tpep_dropoff_da
tetime
"""

df = client.query(query).to_dataframe()
```

Number of rows where pickup time is greater than drop off: 18

3. Analyze rows that have 0 or less passenger count.

```
query = """
SELECT * FROM `nyc-taxi-265120.NYC.2018firsthalf` where passenger_count <= 0
"""

df = client.query(query).to_dataframe()
```

4. Finding the number of trips where the trip duration is more than 12 hours.

The below big query helps us to find out the outlier where the trip duration is more than 12 hours as the maximum trip a taxi can make in NYC should not be more than 12 hours.

```
query = """
SELECT tpep_pickup_datetime, tpep_dropoff_datetime, passenger_count, trip_distanc
e,PULocationID, DOLocationID,fare_amount FROM `nyc-taxi-265120.NYC.2018firsthalf` WHERE T
IMESTAMP_DIFF(tpep_dropoff_datetime,tpep_pickup_datetime,HOUR)>12
"""
df = client.query(query).to_dataframe()
```

Thus BigQuery helps to query the Big Table and return the answer in a matter of seconds. I did the same for 2018 second half, 2019 first and second half respectively.

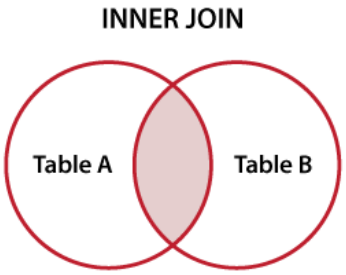


Figure 28: INNER Join in Table A and Table B

Also, I made use of INNER JOIN to eliminate all the out of bounds pickups in New York City. The query for the same is as follows: Table A- Taxi Trip Data and Table B- Zone Data INNER JOIN on zone_id column gives only the zones in NYC city.

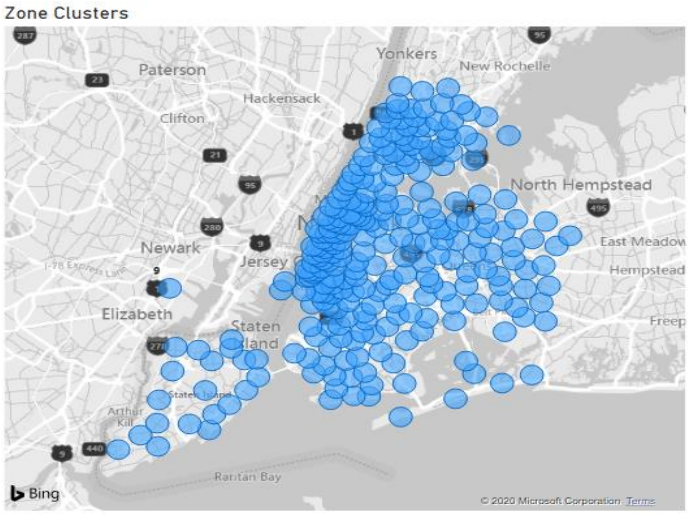


Figure 29: Zone centroid obtained using Power BI

Data Wrangling:

Data wrangling is also referred to as data munging. The main motto is to transform data into a format that is more appropriate and considered more valuable. This means finding relationships among variables to achieve better information about the data.

2018 Data:

Data Wrangling Using Big Query and then Pivot table:

```
query = """
with td as(
WITH AB AS
(
SELECT  CAST(tpep_pickup_datetime AS DATETIME) AS PICKUP,* FROM `nyc-taxi-265120.NYC.2018firsthalf` where total_amount>0 and  tpep_pickup_datetime < tpep_dropoff_datetime AND  TIMESTAMP_DIFF(tpep_dropoff_datetime,tpep_pickup_datetime,HOUR)>12
)
SELECT  PICKUP,CAST(EXTRACT (YEAR from PICKUP) AS STRING) AS year,EXTRACT (DAYOFYEAR from PICKUP) AS daynumber,EXTRACT (HOUR from PICKUP) AS hour, cast(PULocationID as STRING) as LOCATION,COUNT(*) AS numtrips FROM AB group by PICKUP,year,daynumber,hour,LOCATION),points AS
(SELECT *,ST_CENTROID(zone_geom) as p FROM `bigquery-public-data.new_york_taxi_trips.taxi_zone_geom`)
SELECT
cast(DATE(td.PICKUP) as STRING) AS A1,
points.zone_id,
CAST(TIME(DATETIME_TRUNC(td.PICKUP, HOUR)) as string) rounded_to_hour,
SUM(td.numtrips) as label,
FROM  td INNER JOIN points ON points.zone_id=td.LOCATION AND
eXTRACT (YEAR from PICKUP) =2018
group by zone_id,A1,rounded_to_hour
"""

df1 = client.query(query).to_dataframe()
```

Binning so that I can get a window of 1 hr for each zone and each date

A1 column is the pickup date. I took the help of pivot and then unstack the pivot table to get the regular 1-hour window for each zone on each pickup day. Finally, Data is prepared in a format that can be modelled and explored easily. Also, the outliers from the Data have been removed.

Before Data Wrangling:

zone_id time	1	10	100	101	102	106	107	108	109	11	110	111	112	113	114	115	116	117	118	119	12	120	121	122
2018-01-01 00:00:00	NaN	NaN	107.0	NaN	NaN	3.0	474.0	NaN	NaN	NaN	NaN	NaN	15.0	243.0	346.0	NaN	59.0	NaN	NaN	2.0	NaN	NaN	NaN	NaN
2018-01-01 01:00:00	NaN	1.0	169.0	NaN	NaN	16.0	661.0	NaN	NaN	NaN	NaN	NaN	65.0	236.0	207.0	NaN	78.0	NaN	NaN	5.0	1.0	1.0	NaN	NaN
2018-01-01 02:00:00	NaN	NaN	210.0	NaN	1.0	9.0	497.0	NaN	NaN	NaN	NaN	NaN	64.0	174.0	217.0	NaN	75.0	NaN	NaN	1.0	3.0	NaN	NaN	NaN
2018-01-01 03:00:00	1.0	1.0	187.0	NaN	NaN	9.0	390.0	NaN	NaN	NaN	NaN	NaN	40.0	134.0	332.0	NaN	56.0	NaN	NaN	5.0	4.0	1.0	NaN	NaN
2018-01-01 04:00:00	3.0	1.0	194.0	1.0	NaN	8.0	283.0	NaN	NaN	1.0	NaN	NaN	28.0	83.0	228.0	NaN	44.0	NaN	1.0	1.0	NaN	NaN	1.0	NaN

5 rows × 258 columns

Figure 30: Pivot Table

	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	PULocationID	DOLocationID	fare_amount
0	2019-07-01 07:04:53	2019-07-01 07:04:53	1.0	0.0	77.0	264.0	0.0
1	2019-07-01 09:10:24	2019-07-01 09:10:24	1.0	0.0	63.0	264.0	0.0
2	2019-07-01 09:56:06	2019-07-01 09:59:07	1.0	0.0	207.0	207.0	0.0
3	2019-07-01 11:33:44	2019-07-01 11:33:44	1.0	0.0	205.0	264.0	0.0
4	2019-07-01 12:47:33	2019-07-01 12:48:34	1.0	0.0	207.0	207.0	0.0

Figure 31: Sample Records

Sample of Dataset after Data Wrangling:

	zone_id	time	pickups
0	1	2018-01-01 00:00:00	0.0
1	1	2018-01-01 01:00:00	0.0
2	1	2018-01-01 02:00:00	0.0
3	1	2018-01-01 03:00:00	1.0
4	1	2018-01-01 04:00:00	3.0

Figure 32: Dataset after Wrangling

The above-wrangled data is joined with zone dataset to get the associated zone coordinates and the name and borough the zone is associated with it.

Data Visualization & Exploration:

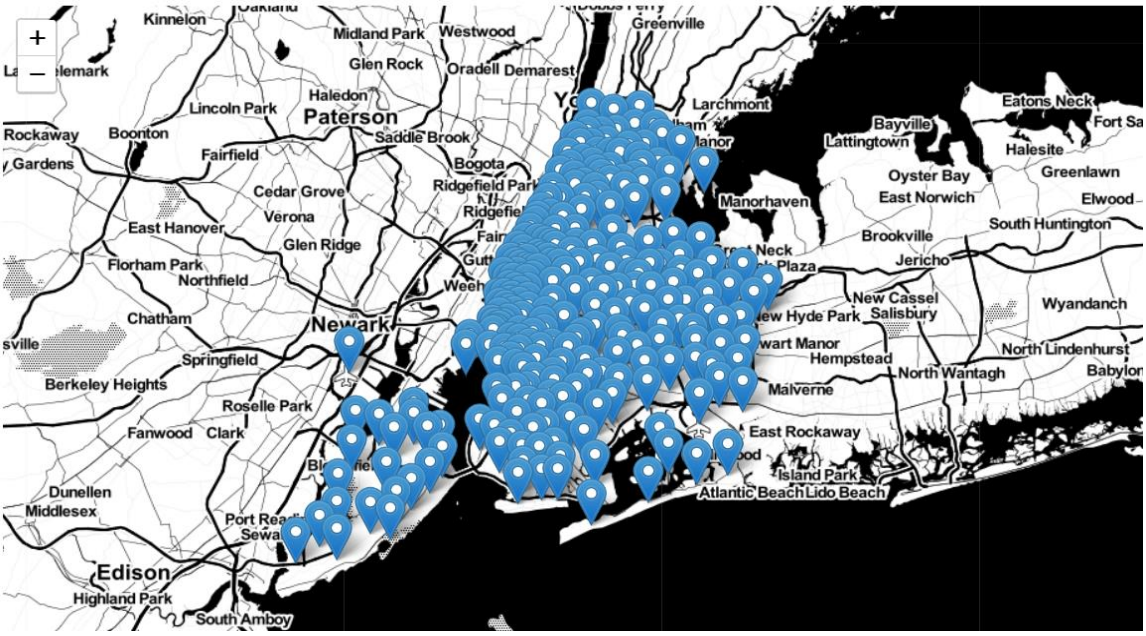


Figure 33: Cluster centers of each zone after data modelling.

Concatenate all the Data together:

```
##_________Concating all Data Together _____________________##

import pandas as pd

concat=pd.concat([first_2018,second_2018,second_2019,first_2019],ignore_index=True)
```

Lastly loading all the cleaned datasets to the Big query Data warehouse.

```
## _________________STORING PREPARED DATA TO GOOGLE CLOUD_____________##

from datalab.context import Context

concat.to_gbq('hello.CombinedData',

              Context.default().project_id,
```

```
        chunksize=500000,
        if_exists='append',
        verbose=False
        )
```

Summarize and Visualize the Data:

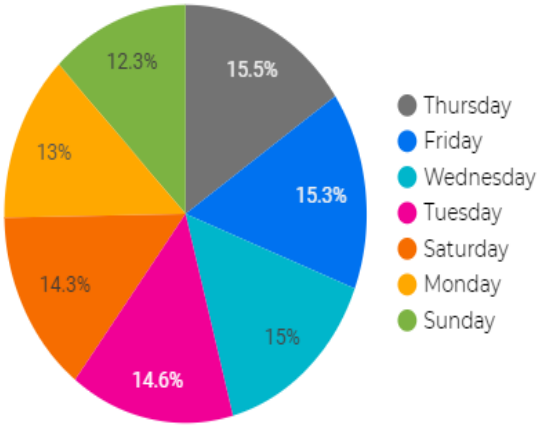


Figure 1 2019

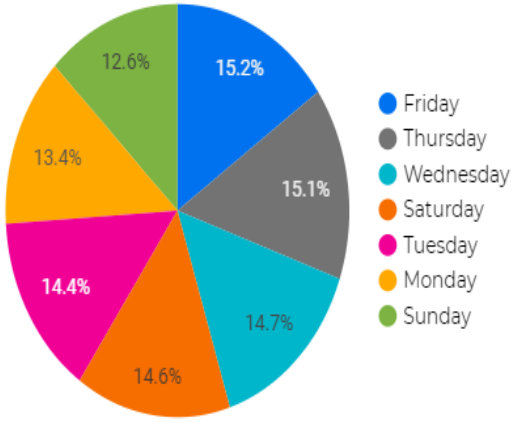


Figure 2 2018

Figure 34: Pie Chart showcasing trips on weekdays in 2019 and 2018

This pie chart gives us the insights that there are more pickups on Thursday and the least on Sunday. This data set was visualized using the 2019 Data Set. Hence the day of the week is also an important feature in building the model. Also, there is not much relation between the weekdays and the weekends. So, this feature must be trained cautiously.

Also, by carrying data exploration, on the 2018 Taxi Data set it can be seen that most of the pickups were done on Fridays as compare to the other days of the week. This step is essential for model planning. Also, I decided to make a line chart depicting the number of pickups month over month to see how the pickups are varying month over moths. Data Visualization is important to discover greater insights from the dataset.

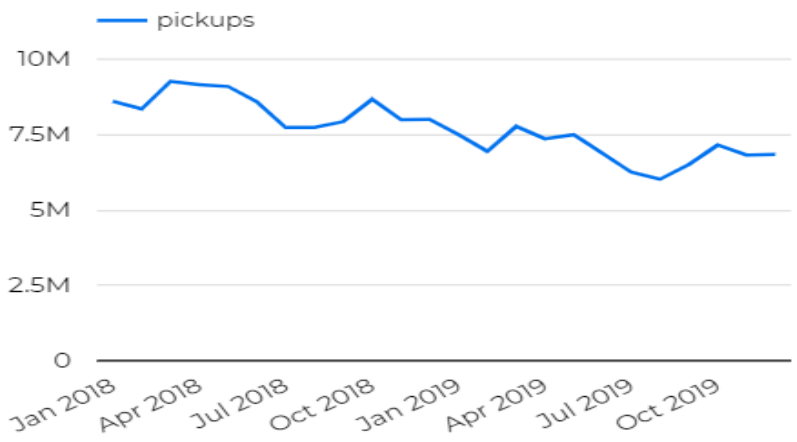


Figure 35: Line Plot showcasing the pickups Month by Month

The above shows the pickups count over the period. There is a gradual drop in the number of pickups overtime for the NYC yellow cab taxi. Also, I visualized how the top 10 zones int 2018 perform in 2019. From the bar chart, it is clear that the Yellow taxi is passenger count is dropping as the competition in the market is increasing.

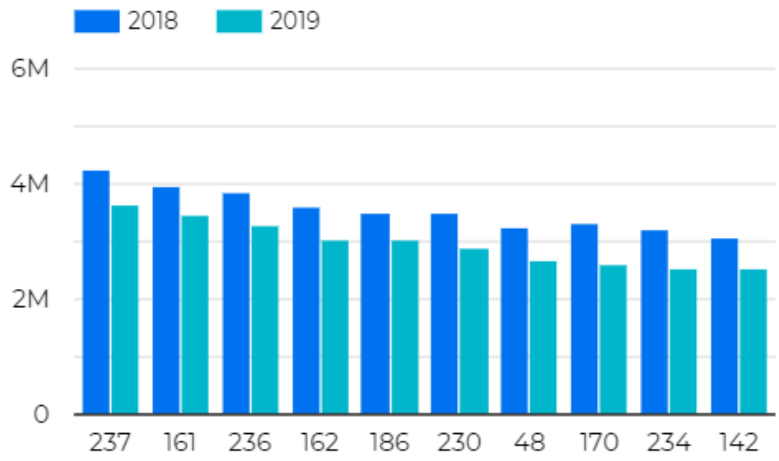


Figure 36: Bar chart showing the Zone ID having high pickups Demand

There is a drop in passenger count in all the top 10 trending zones in 2018. I think my model will help the NYC yellow cab taxi drivers know in advance the passenger they are expected to get in a particular zone depending on my model. I also used Plotty in python to visualize how pickups change by day over a day in 2018 and 2019. The figure is depicted below:

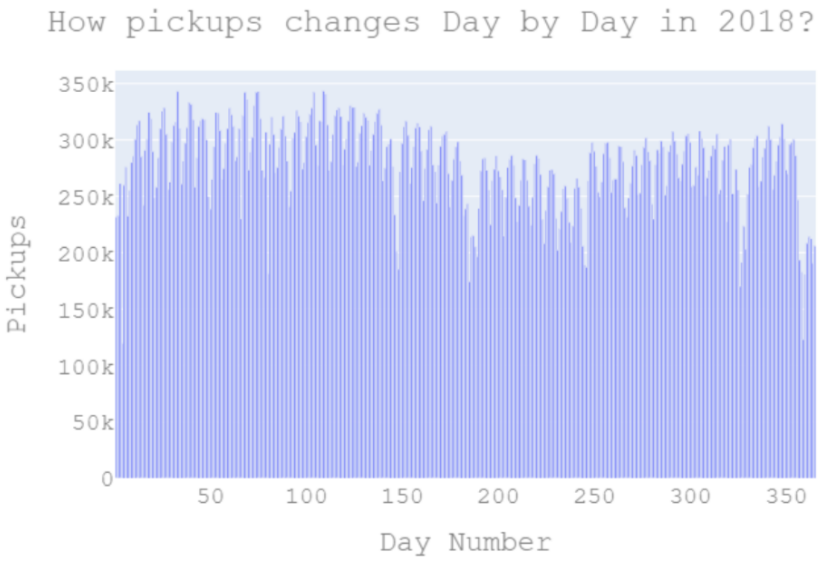


Figure 37: How pickups change Day by Day in 2018

The figure above depicts the pickups variation day by day in 2108. The below figure depicts the variation in pickup day by day in 2019.

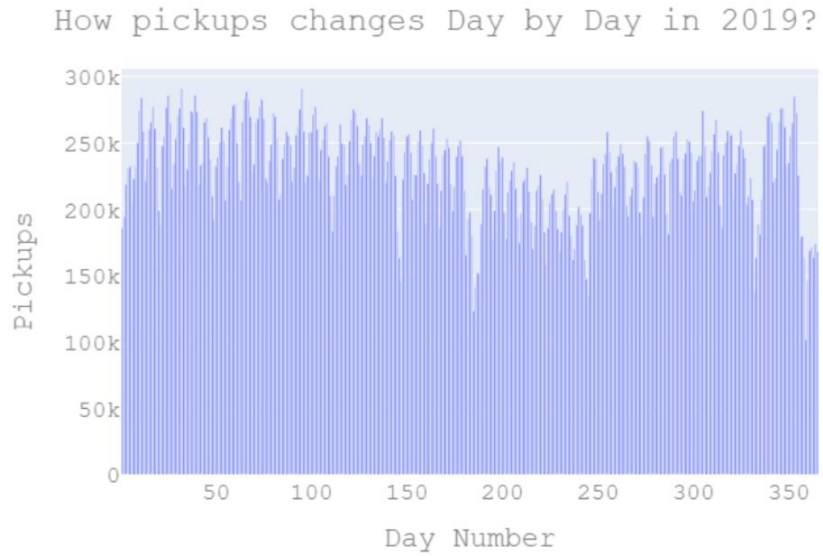


Figure 38: How pickups change Day by Day in 2019

The above two bar plots give a distribution of pickups day by day in 2018 and 2019. This gives us an idea the pickups count in 2019 is below the number of pickups in 2018.

6.3 Model Planning:

In model planning, I have to determine the relationship between the variables so that I can perform variable selection. Also, I have to research and find a suitable model that can give me optimum results.

It includes two important steps:

1. Variable selection
2. Model Selection

Variable Selection

Data Exploration is the phase where one tries to know the data and how the, unlike variables, interrelate with each other. In Machine Learning, Data Exploration always precedes the formation of the predictive model as it consents us to come up with better ideas to increase the models' performances.

Feature Engineering

In model arranging, the strategies, methods, and work processes are being determined. During this phase, the dataset is explored to learn about the relations amongst different variables and then selecting the key variables and for the most suitable models.

Performed One Hot Encoding:

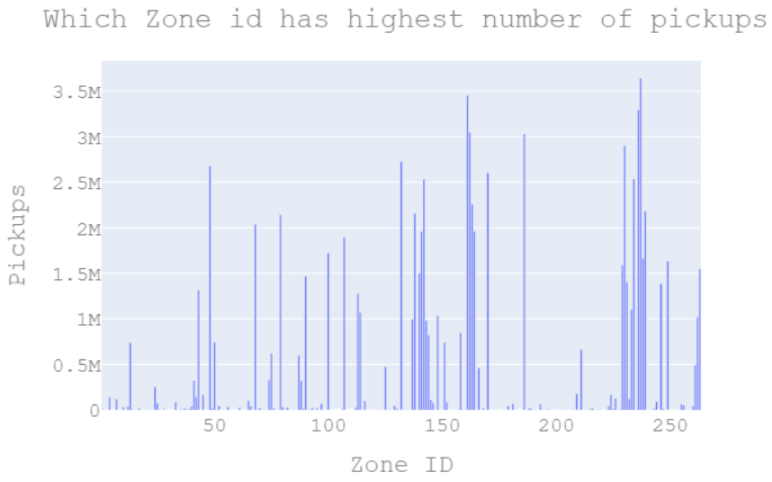


Figure 39: Which Zone ID has the highest number of Pickups

The zone_id is a categorical variable. One hot encoding can be used to align zone_id column-wise.

I performed One Hot Encoding using the pandas to get dummies function.

```
#_____ONE HOT ENCODING____

import pandas as pd

one_hot = pd.get_dummies(data['zone_id'])


data = data.drop('zone_id',axis = 1) # Join the encoded df data = data.join(one_hot)
```

It is a technique that helps to handle categorical variables. We cannot use label encoding in this case as zone 1 and zone 2 does not signify that 1 is greater than 2. Hence the approach of one hot encoding was followed.

DateTime Column Feature Extraction

This dataset for models other than ARIMA. ARIMA was tested with the inbuild DateTime feature as ARIMA can handle the DateTime feature automatically. But for other regression techniques, I decided to add an hour, day of the week, day number, min and max temp in the Data and scaling it so that it can be trained using other regression techniques.

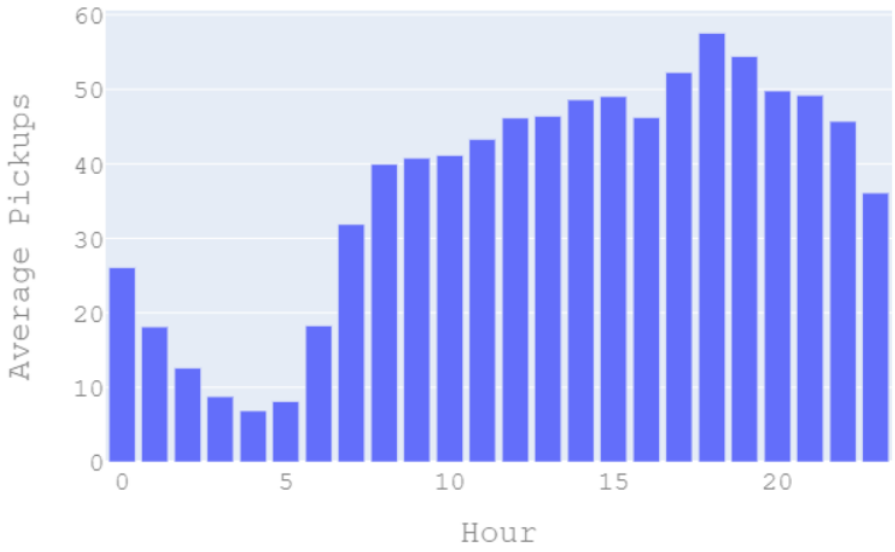


Figure 40: Average Pickups Hour by Hour 2019

The average pickups per hour in 2019 shows that hour is also an important factor to consider while building the model. Data exploration is always been beneficial to get an important variable and make a variable selection.

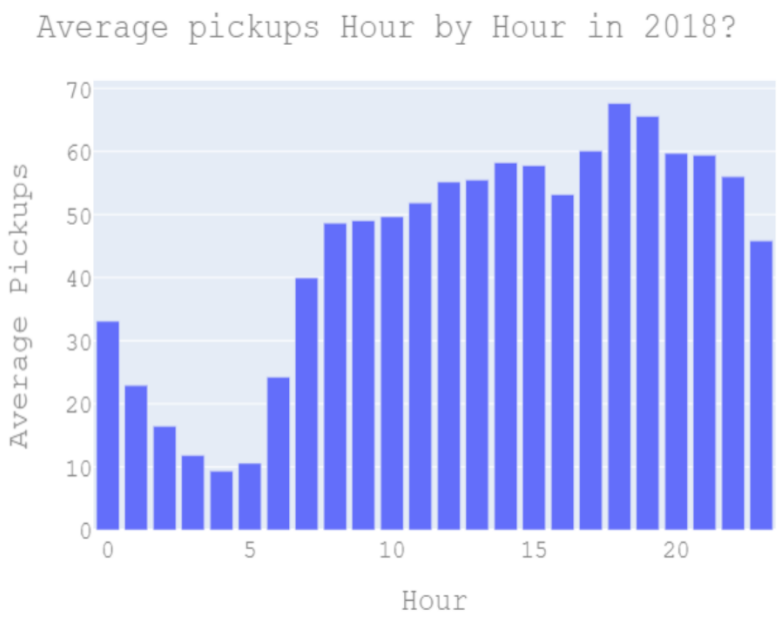


Figure 41: Average pickups Hour by Hour in 2018

By exploration, it can be said that the variables to be considered for training a model are as follows:

1. **Zone id:** Zone id is a categorical variable and can be thus has been handled using the technique of One Hot Encoding.
2. **Hour of the day:** As we say above that the rides drop between 1 AM - 5 AM. When it is 6 AM the number of pickups increases and we expect the maximum pickups between 6 PM to 9 PM. Hence it is also an important factor
3. **Day of the week:** It is again a feature that can be considered to train the model. Like on weekends the pickups will be different as compared to the weekdays.
4. **Day of the year:** It is again a feature that can be considered to train the model. The pickups vary as the days progresses.
5. **Min Temp**: It can be considered to build the model. The pickups vary min temp changes.
6. **Max Temp:** It is the maximum temperature that can be reached in a day It is also considered one of the features while building the model.

These were some of the important variables to be considered for the model building phase.

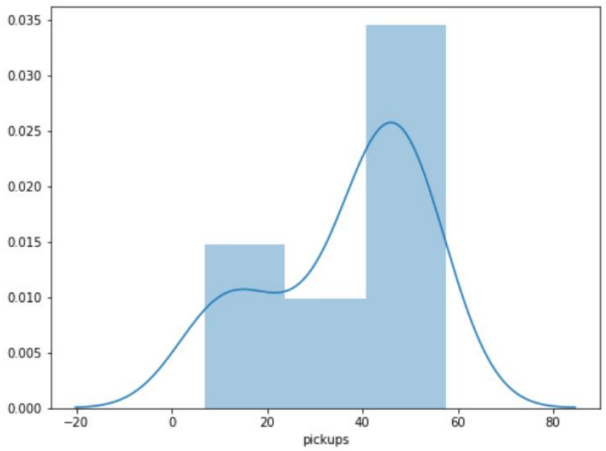


Figure 42: Distribution plot using Seaborn

The Correlation Matrix: It helps to study the relationship between the variables. Using the correlation matrix it is easy to know which variable have high correlation with other variables. If that's the case we can easy remove the one of that variable and keep only one. Thus I decided to plot the correlation matrix between the variable I thoroughly explored. The value in the matrix varies between +1 and -1. If the value is close to ± 1 then it denotes that the variables are highly correlated and one of them can be easily dropped. So exploring the correlation matrix given below we can state that hour can highly determine pickups as compared to other variables. Also zone id is a categorical variable. Thus it was not a part of this matrix. But that is clear that Zone Id is also an important feature to consider.

Figure 43: Correlation matrix

Model Selection:

Classification, Regression • Association rules • Text/Image/Video Analysis

The problem is to predict the number of pickups in different zones in NYC city. The problem will be categorized among the regression problem. Though the data also gives the ability to apply time series analysis. Therefore the models that will be considered for my project :

Time Series Analysis using ARIMA. Regression using Linear Regression, Random Forest Regression, XGboost Regression, Light GBM, etc. I applied all the models and calculated the MAPE, MAE, and MASE for the same to see which model performs the best. I choose the best model from all the given above.

Time Binning

Also, Using the resample function we can divide the time into bins. I have taken a bin of one day. To get the pickups in a day. Later if the model is accurate then I will divide the bins to smaller bins of the hourly window.

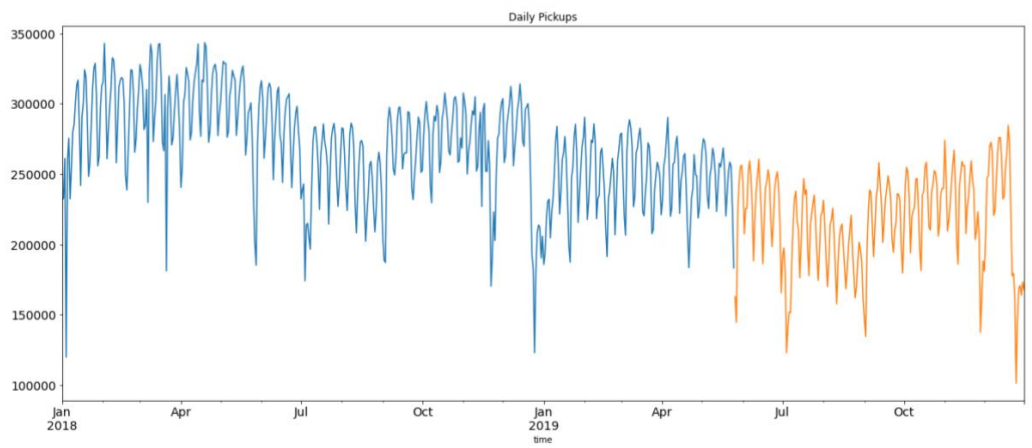


Figure 44: Time Binning Day by day

ADF Statistic: -1.8620831215235754

p-value: 0.35010313377312674

Critical Values:

1%: -3.439593802573824

5%: -2.865619356068967

10%: -2.568942332870462

The p-value is not in 0.05. So, the seasonality from the Data needs to be removed. This is done in the Model Building Phase.

Summary

Now I have completed feature engineering both for ARIMA and also for Xgboost, RandomForest, and other regression techniques. Now I will apply cross-validation and use MAE, MASE, MAE, R2, RMSE as the benchmark for deciding the model which I will be using in my Flask application.

Variable selection: For the regression method, I selected the following: Zone id: Zone id is a categorical variable and can be thus has been handled using the technique of One Hot Encoding. Hour of the day: As we say above that the rides drop between 1 AM - 5 AM. When it is 6 AM the number of pickups increases and we expect the maximum pickups between 6 PM to 9 PM. Hence it is also an important factor. Day of the week: It is again a feature that can be considered to train the model. Like on weekends the pickups will be different as compared to the weekdays. Day of the year: It is again a feature that can be considered to train the model. The pickups vary as the days progresses. Min Temp: It can be considered to build the model. The pickups vary min temp changes. Max Temp: It is the maximum temperature that can be reached in a day It is also considered one of the features while building the model. For time series analysis I selected the following: Time and pickups after performing Binning.

Model Selection: For the regression method I selected the following:

1. Linear Regression
2. xgboost
3. Random Forest Regressor
4. Light GBM

For time series analysis I selected the following:

1. ARIMA
2. Seasonal ARIMA

3. Prophets

MAE, MASE, MAE, R2, RMSE are some performance measures that are considered in deciding the final model.

6.4 Model Building:

The key activities of this phase is to split the data into train and test and also perform cross validation to prevent overfitting or underfitting. Different model that were selected in the previous phase were used and Grid Search CV was used to select the optimal hyperparameters for the selected models. Lastly the Regression Error Metrics of MAE,MASE and R2 were used to determine the optimal model to predict the taxi pickups.

But for time series forecasting we can consider .7 of the data can be considered for training and .3 for testing. But the data should be ordered in a time series analysis. So, I have to divide data as shown below and make it ordered as shown:

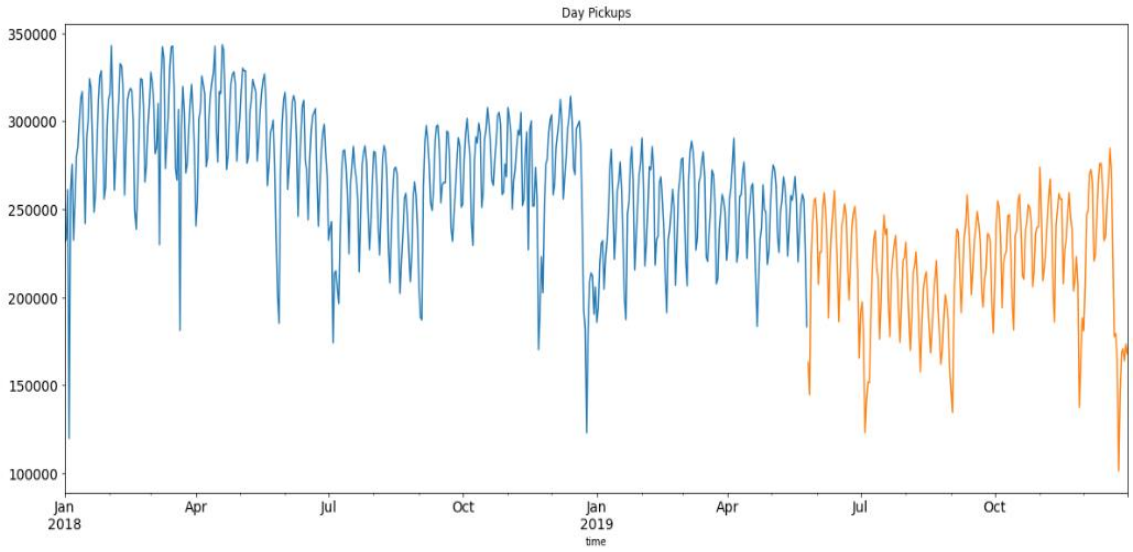


Figure 45: Train, Test and Validation Split

From the dataset, I got the mean and SD(standard deviation) to get an idea of how the time series data varies over time. This was done to get an idea of how the data varies during the course of time.

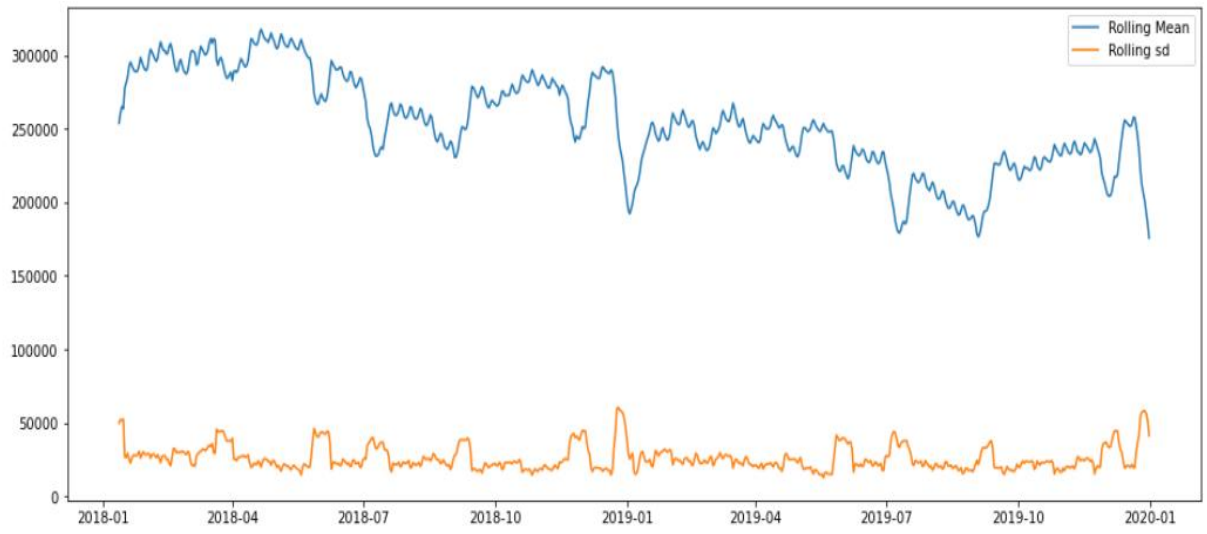


Figure 46: Exploring Rolling Mean and Standard Deviation

In time-series data, it is very important to get the trend and seasonality out of the data. By using the python library I was able to get the trend, seasonality, and residual of the given dataset.

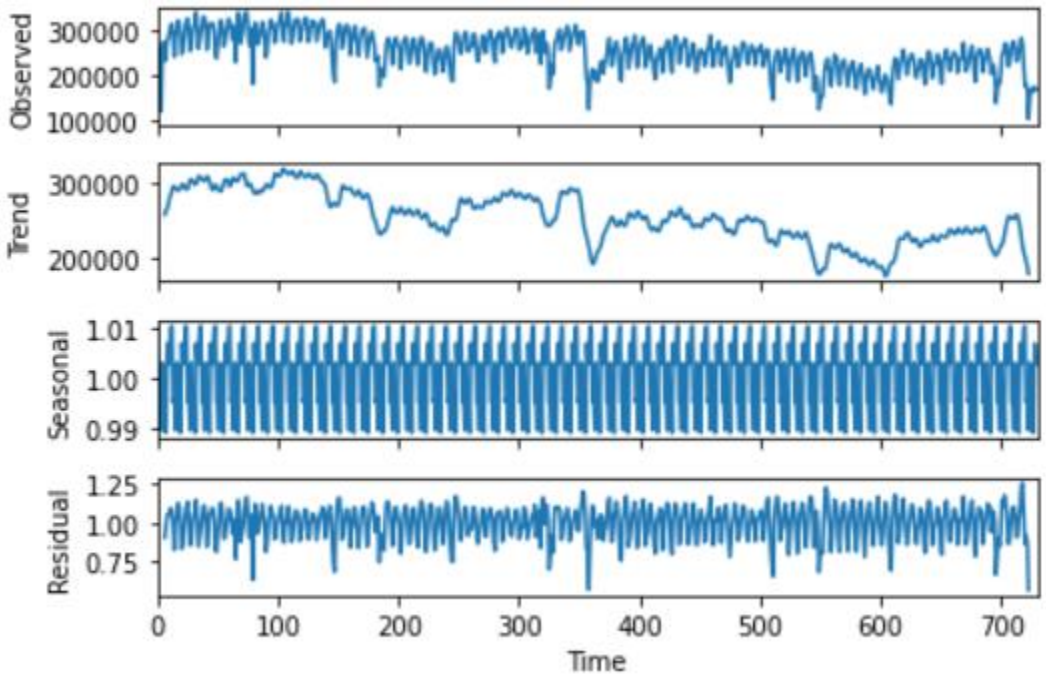


Figure 47: Getting Trend, Seasonality, Trend and Residual

Now I removed the trend and seasonality from the data and got the following :

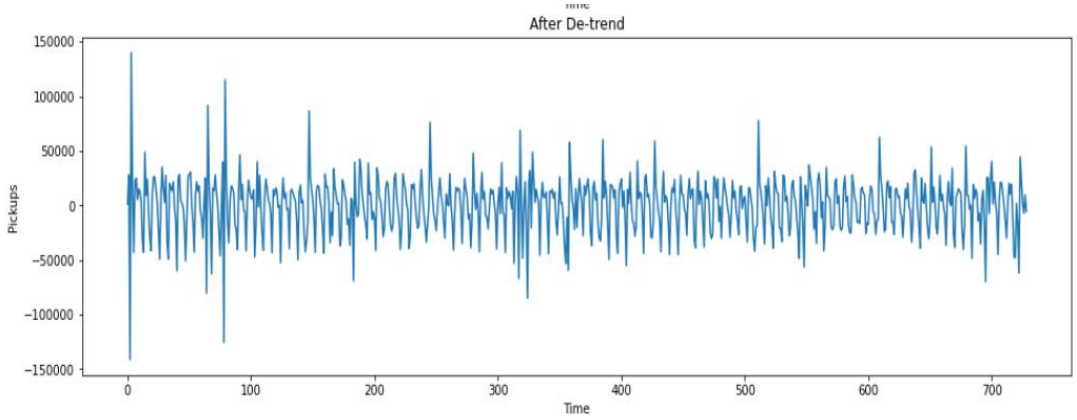


Figure 48: After removing Trend

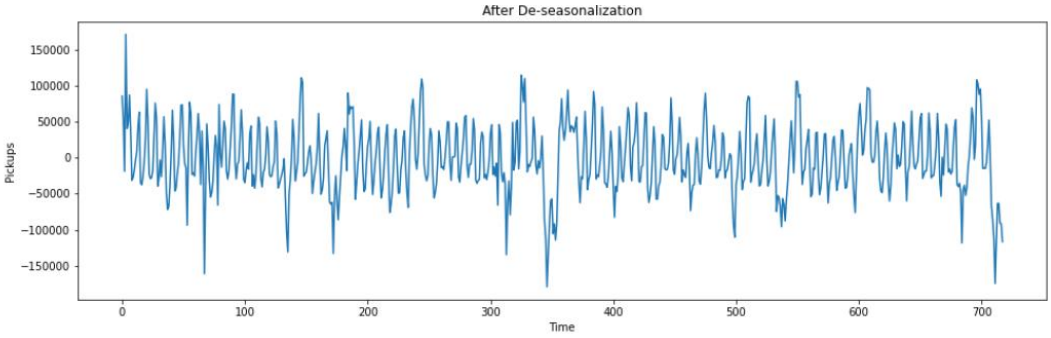


Figure 49: After removing Seasonality

After removing the trend and seasonality I did the Dickey-Fuller test to test for seasonality of them and then decided to forecast using the auto ARIMA model.

```
Results of Dickey-Fuller Test:
Test Statistic                -7.981551e+00
p-value                        2.615200e-12
#Lags Used                     2.000000e+01
Number of Observations Used    6.970000e+02
Critical Value (1%)           -3.439767e+00
Critical Value (5%)           -2.865696e+00
Critical Value (10%)          -2.568983e+00
dtype: float64
```

Figure 50: Result of Dickey-Fuller Test

p-value, in this case, is equal to 2.615200e-12. Hence we can assume the NULL hypothesis.

The p-value is < 0.05. This means the seasonality from the data has been removed and the dataset is now considered stationary. Now I can forest the stationary dataset and then add back trend and seasonality to the data points.

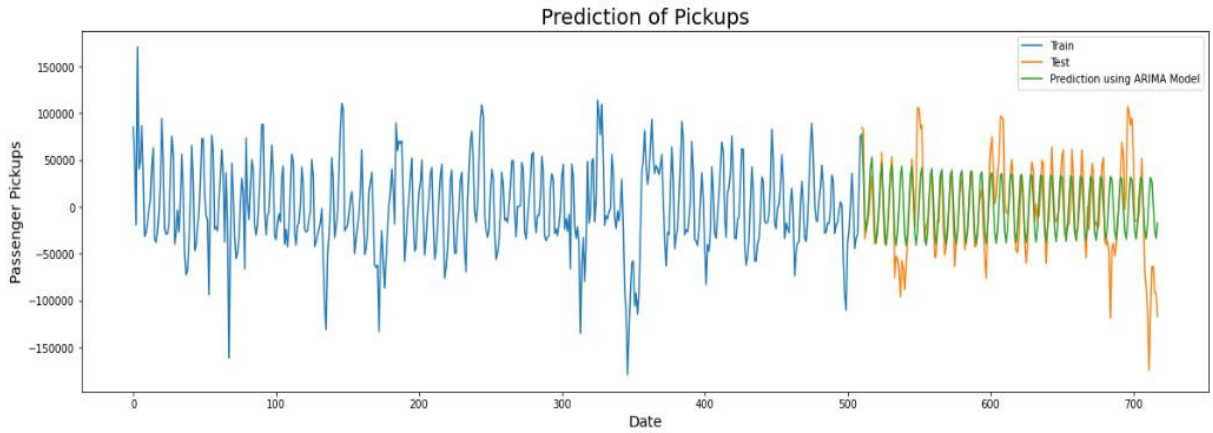


Figure 51: Auto Arima Forecasting

Then adding back trend and seasonality I got the MAE of 17.7 %. I have also used Prophet for forecasting I added the trend and seasonality back to the data and got the following output :

Figure 52: Prophets Time-series Exploration

Moreover, I got the following results from forecasting. Prophets library gives the ability to view how data varies across different seasons and also provides the ability to see the trend in the dataset. I decided to further pursue with SARMIA as I collected how the seasonality varies in the dataset sign Prophet library.

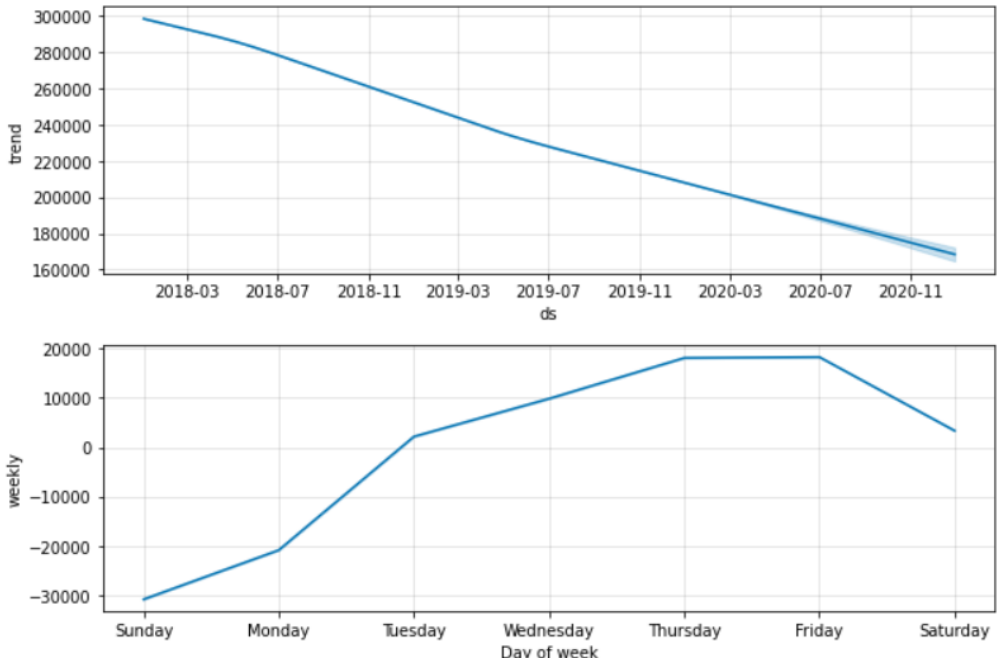


Figure 53: Getting the weekly trend

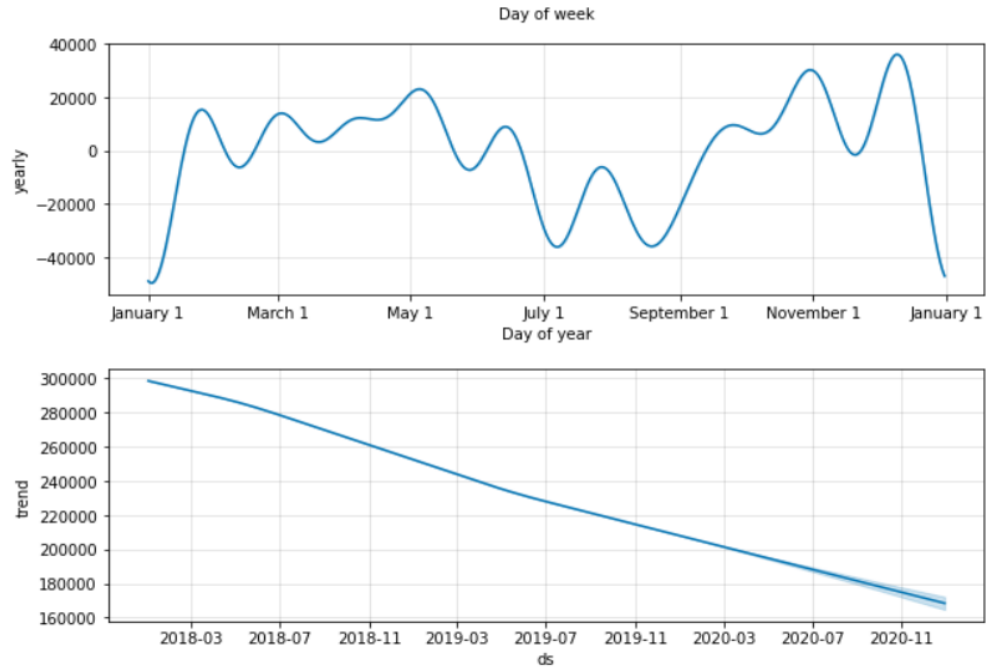


Figure 54: Knowing the trend and seasonality yearly

The error involved using ARIMA was more. So I decided to move on with seasonal ARIMA as I got to figure out using Prophet that there is a seasonality involved in the data.

```
def optimizeSARIMA(parameters_list, d, D, s):
    results = []
    best_aic = float("inf")

    for param in tqdm_notebook(parameters_list):
        # we need try-except because on some combinations model fails to converge
        try:
            model=sm.tsa.statespace.SARIMAX(ads.Ads, order=(param[0], d, param[1]),
                                            seasonal_order=(param[2], D, param[3], s)).fit(disp=-1)
        except:
            continue
        aic = model.aic
        # saving best model, AIC and parameters
        if aic < best_aic:
            best_model = model
            best_aic = aic
            best_param = param
        results.append([param, model.aic])

    result_table = pd.DataFrame(results)
    result_table.columns = ['parameters', 'aic']
    # sorting in ascending order, the lower AIC is - the better
    result_table = result_table.sort_values(by='aic', ascending=True).reset_index(drop=True)

    return result_table
```

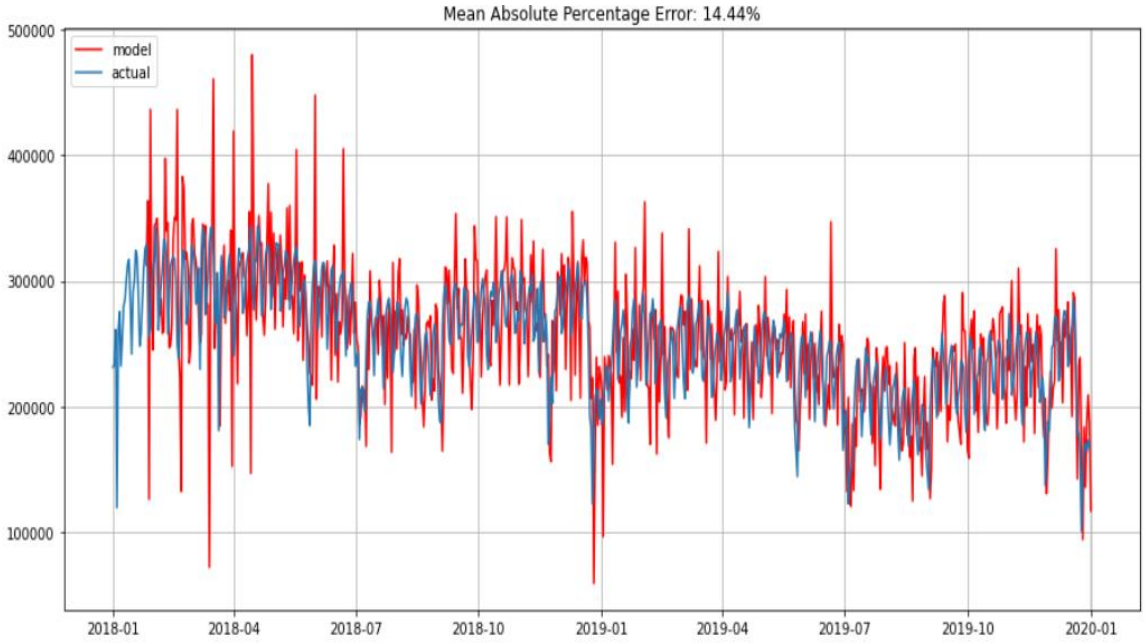


Figure 55: SARIMA Forecasting

Splitting of Data set into test and train and validation (I have performed cross-validation to pick the best model for predicting the number of pickups):

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=20)
```

The feature that is considered importance using Xgboost are as follows:

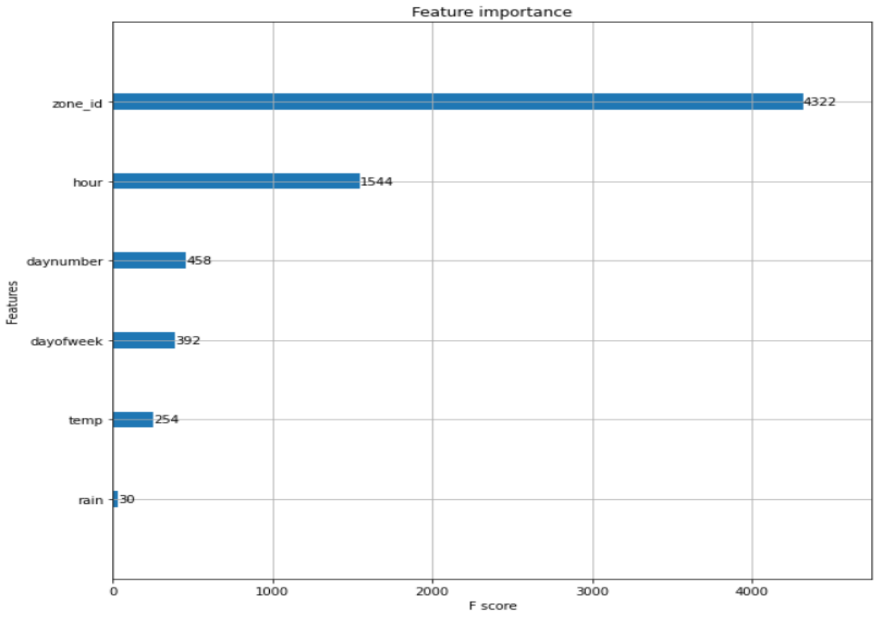


Figure 56: Xgboost Feature importance

Xgboost with GRID SEARCH CV:

```
from sklearn.model_selection import GridSearchCV

from sklearn.ensemble import XGBRegressor
```

```
param_grid = {
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
# Create a based model
rf = xgb.XGBRegressor()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid, cv = 3, n_jobs = -1, verbose = 2)
```

RANDOM FOREST with GRID Search CV :

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
# Create a based model
rf = RandomForestRegressor()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 3, n_jobs = -1, verbose = 2)
```

The use of Light GBM and Linear Regression but got an unexpected result as shown below:

```
import numpy as np

def MASE(training_series, testing_series, prediction_series):

  n = training_series.shape[0]

  d = np.abs(  np.diff( training_series) ).sum()/(n-1)

  errors = np.abs(testing_series - prediction_series )

  return errors.mean()/d
```

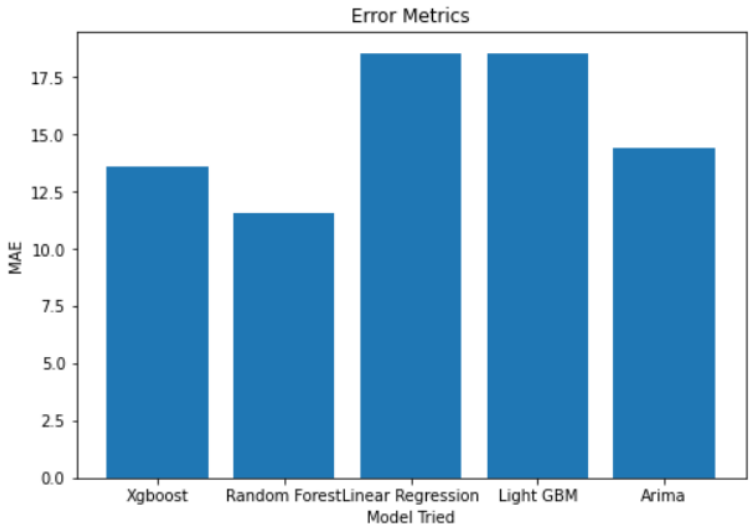


Figure 57: MAE of all the models

Further exploration can be seen before selecting the actual model between Xgboost and Random forest for achieving precise, faster and better results:

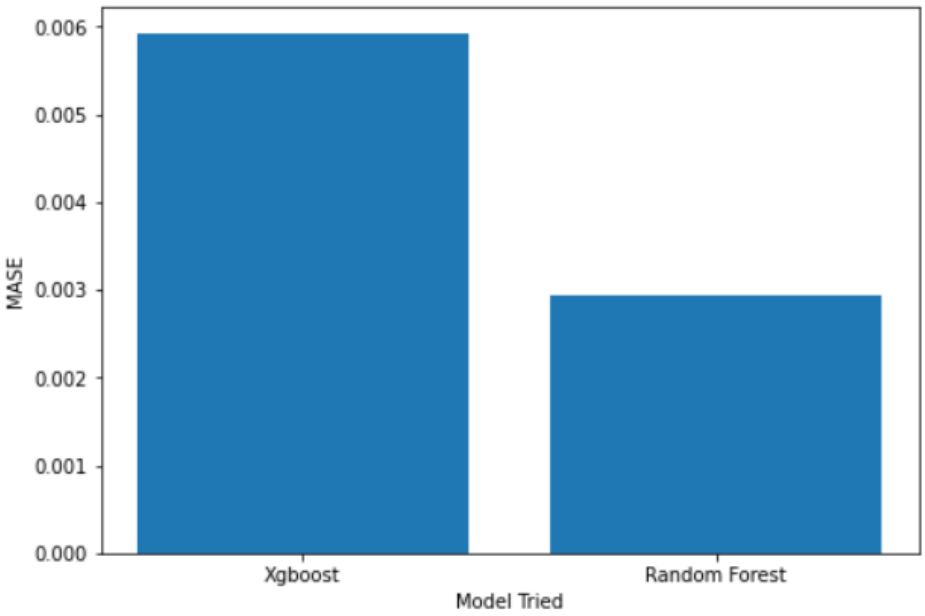


Figure 58: MASE OF all models

6.5 Communication Result:

Dashboards are prepared using Data Studio that is easy to share between executives and also helps the executives to take decisions effectively. The model used in my web application is Random Forest Regressor due to its least error rate measured using MAE, R2, MASE, etc. Also, Dashboards were an important part of the communication phase.

The initial plan was to use Power BI but it would have costed me for sharing my reports with other executives and it is not effective as much as the data studio in terms of convenience. Hence I decided to proceed with Data Studio which provides this ability free of cost. The following results are determined and using the model used for the project and data training and communicated using

Dashboards:

- Comparison of Demanding Areas in the year 2018 and 2019.
- Forecasted weekly pickups in the upcoming weeks.
- Will help the TLC to make an effective business decision.
- Also, help the taxi drivers to discover the important areas in NYC city to get the passengers.

Yearly Comparison Dashboard:

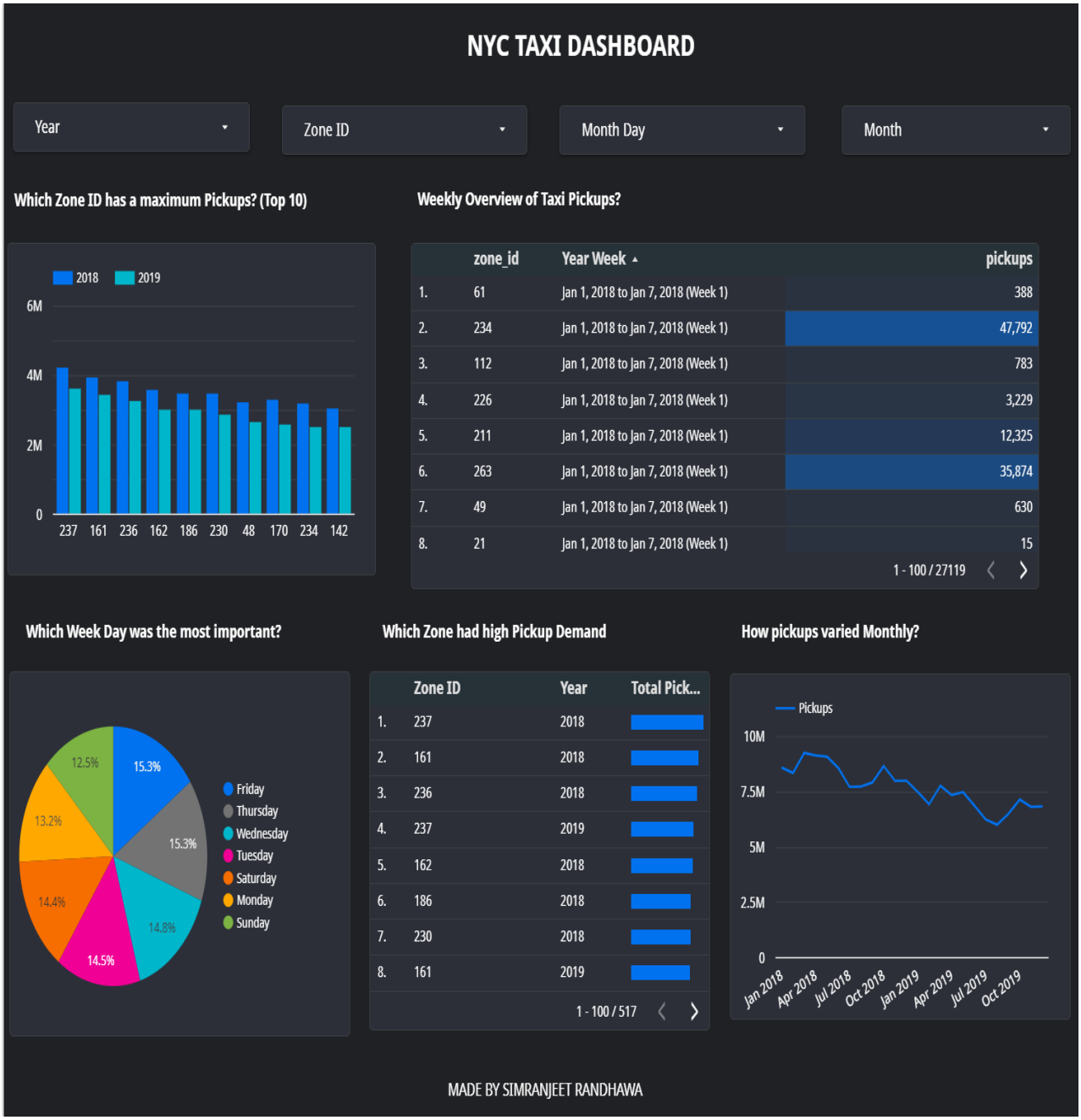


Figure 59: Dashboard of 2018 and 2019 yellow NYC taxi Dataset

Recommendations :

The visualized reports are enough to provide the knowledge to the user to provide sufficient information regarding the upcoming weekly forecast for the demand of pickups in NYC city. The

following Dashboards show the exact number of taxis required to be sent in each zone to meet the passenger's demand and grow the business.

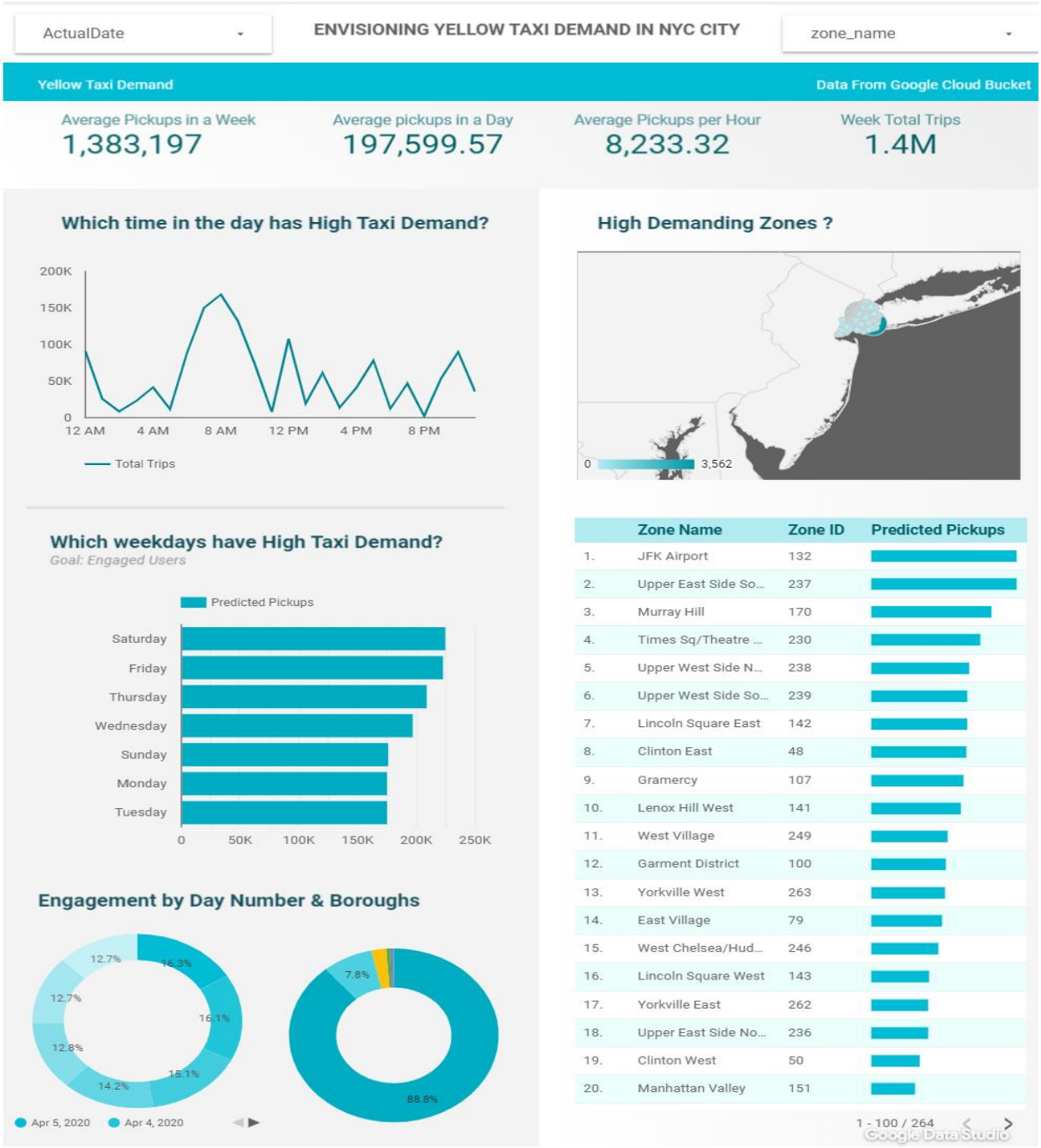


Figure 60: Weekly Forecasted Trips Dashboard

Next Step: The Dashboard can be used by the executive to know whether the current demand is met or not.

• Model Selected: Random Forest Regressor because it showed the least error as compared to other techniques.

Model	MAE	R2	RMSE
Random Forest Regressor	11.57897810134032	0.45528366614765486	24.93736930641297
Xgboost	13.57897810134032	0.7528366614765484	32.93736930641297
Light GBM	18.57897810134032	0.9552836661476548	40.93736930641297
ARIMA	14.95411850134032	0.81357330134032	35.56811210134032
Linear Regression	21.57677810134032	0.9972836661476548	43.93737930641297

Figure 61: Error Measure for all the Tried Models

• Code: The code for the entire project can be found at https://github.com/ssrbazpur/Envisioning-Yellow-Taxi-High-Demand-Areas-in-NYC-city

• Technical specifications for implementing the code in dev/production environment: The code has been deployed on the GitHub. Soon technical documentation will be updated. I am managing the website for the project on the Google Cloud Platform.

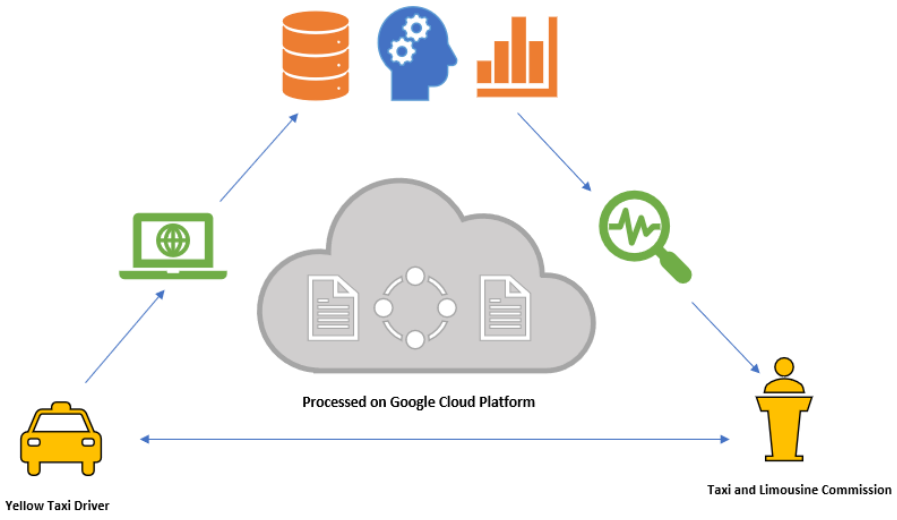


Figure 62: How the Application Works

The taxi driver can use the website to know the predicted passenger in any zone of NYC city. The TLC can use the dashboards to ensure that the demand in each zone of NYC city for the yellow taxi is met by monitoring the yellow cab taxis.

6.6 Operationalize:

What is Flask?

Flask is a web development framework that is pythonic which includes several modules and collection of various libraries which helps easy development of any web application. The Flask framework uses WerkZeug as it's base that handles the data response, request and other utility functions and implements it using the WSGI toolkit. It is also known as a "micro-framework" because it supports any built application to simple and yet extensible. Further, it does not have any abstraction layer to handle the database and not even the validation support for any type of form.

Steps to build docker images and host it on Google-Cloud platform:

1. To set up Docker image: (make sure to replace [PROJECT_ID] with your cloud project ID.)
2. Clone Repository: git clone the repository.
3. Go to Source Directory: cd source directory
4. The docker image can be using docker build -t gcr.io/[PROJECT_ID]/source file.
5. Configure Docker for Google Cloud: gcloud auth configure-docker
6. The image can be pushed using : docker push gcr.io/[PROJECT_ID]/source and that's it!

Website View:

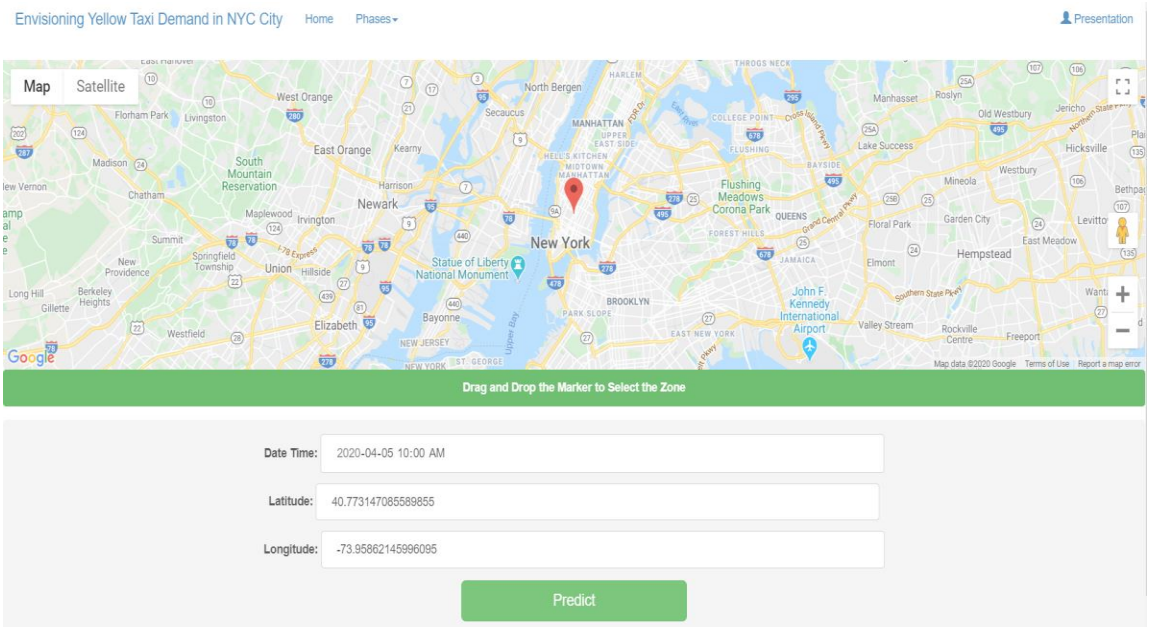


Figure 63: Home Page of the Website

As seen in the above screenshots of the website the user can pick the location on the map along with selecting a date and time and hit the predict button the result will be shown on the next page as below with the number of picks in the zone selected.

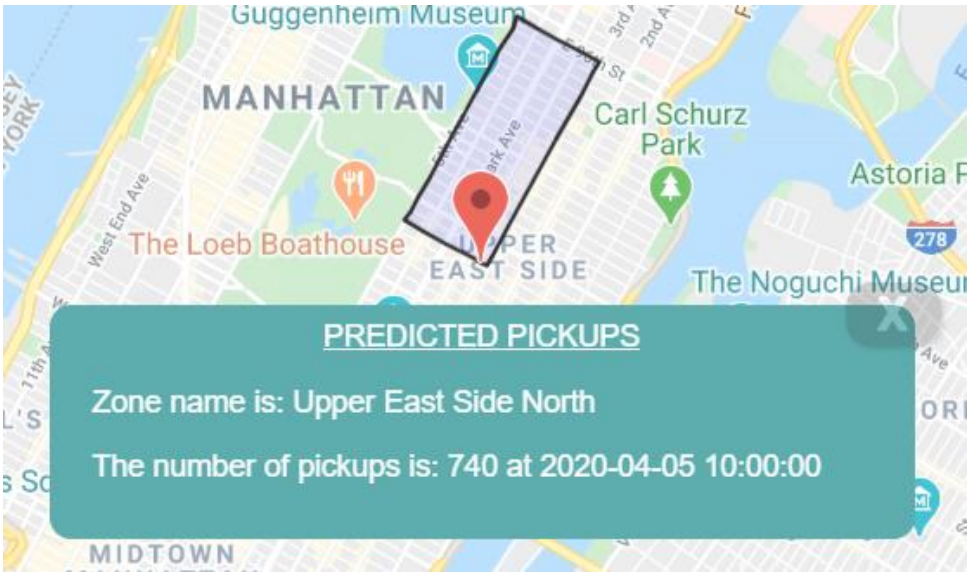


Figure 64: Predicted Pickups

There will be some more details about the location like borough name, zone id, current time and weekly temperature of New York City. But the user will have more detailed options to check the reports of the Yellow Taxi pickups which are "**Visualizing a weekly forecast**" and "**Envisioning 2018 & 2019 reports**". These two choices will provide the user with enormous information about the scope of the business and the future number of pickups.

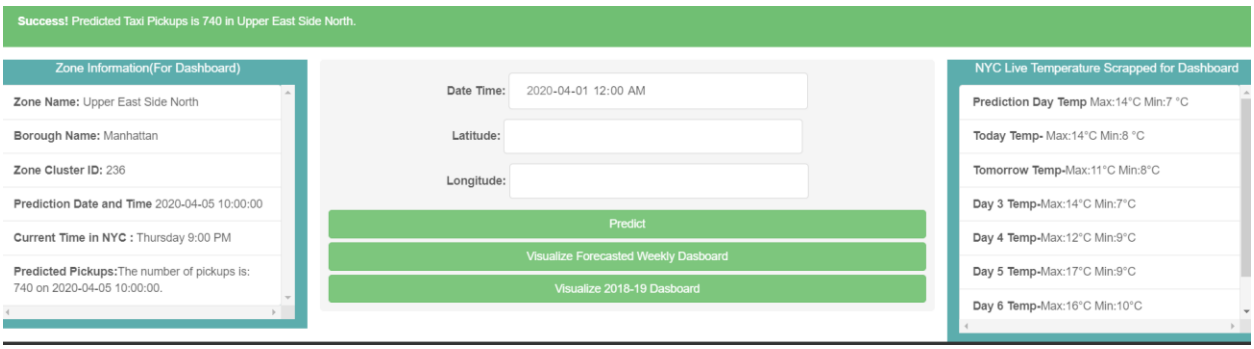


Figure 65: Three Buttons on the Bottom

So considering the first option of Visualizing the weekly forecast the will be able to explore the following details on the dashboard. The details like which time in a day there is the highest

demand? Which weekdays are the busiest? So that it can help the driver pick more customers and that leads to the benefit of the Yellow Cab services.

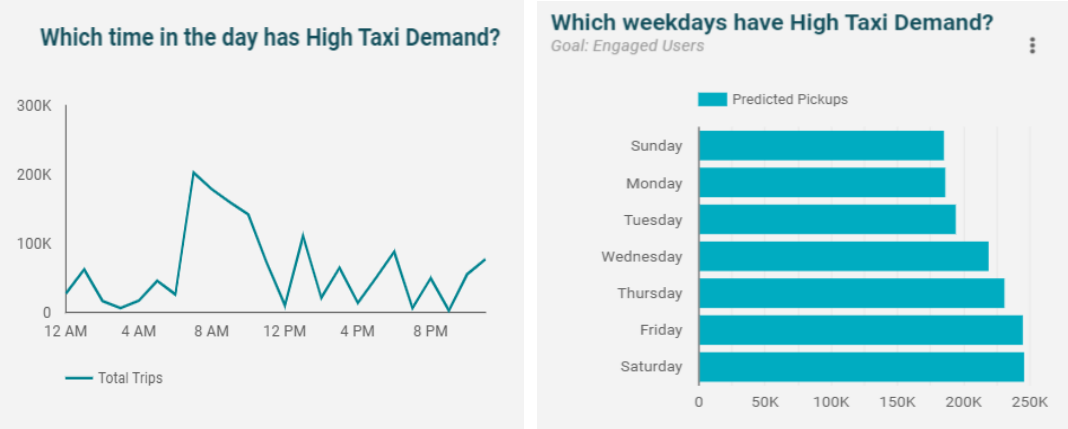


Figure 66: Three Buttons on the Bottom

There are two filters available for weekly dates and zone name in case if the user wants to change the zone and look details for the particular area only. Also, these provide details like the average number of pickups during the hour, a day, and a week which can provide huge benefits for the business of the cab service.

Figure 67: All metrics on the Forecasted Dashboard

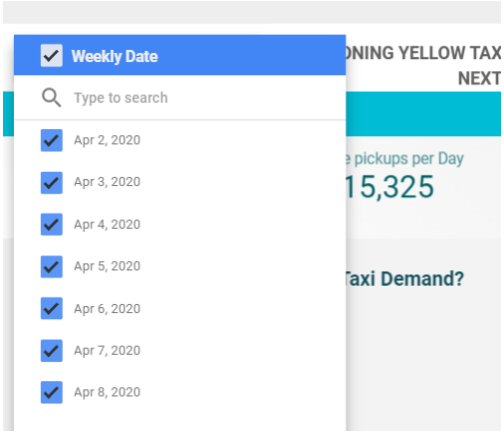


Figure 68: Weekly Data filtering operation

Further, if the user selects the "Comparison reports for the year of 2018 and 2019" the dashboard provides a major amount of details about the pickups for the days, zone and so on. For which there are four (4) filters available which can be seen in the below snapshot as Year, Zone ID, Month Day and Month.

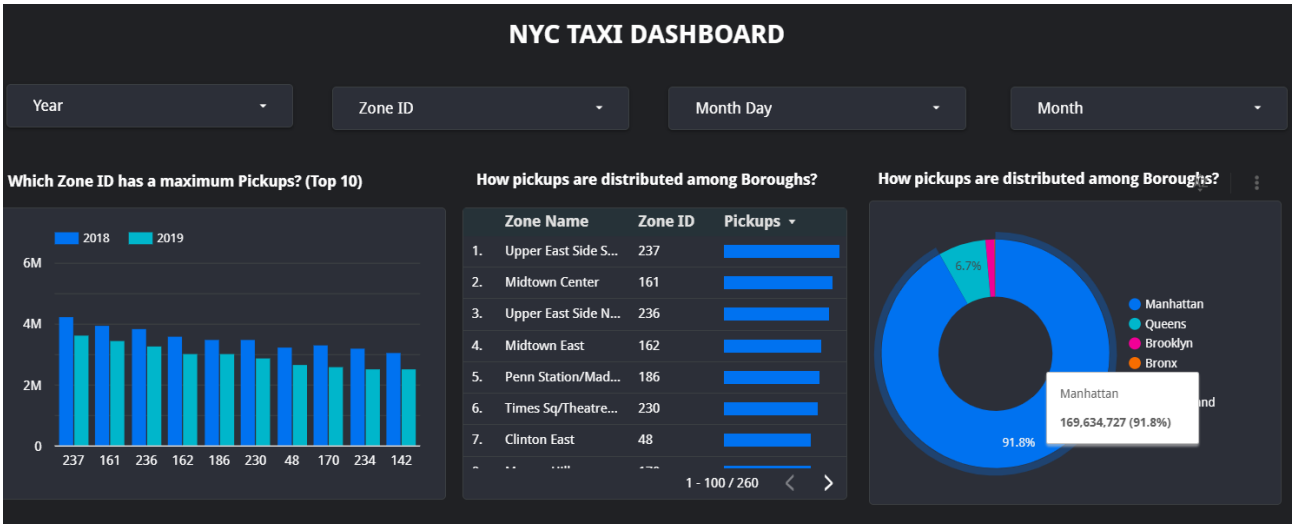
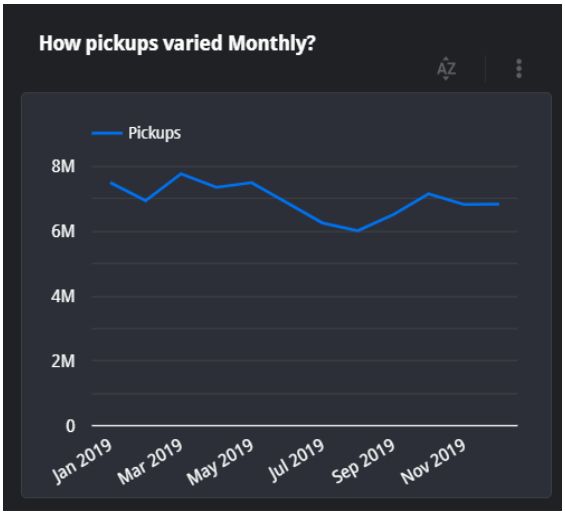


Figure 69: Hover to see the Result of Boroughs

The user can filter the data using the available options and gain good knowledge to achieve better profit. Such as:



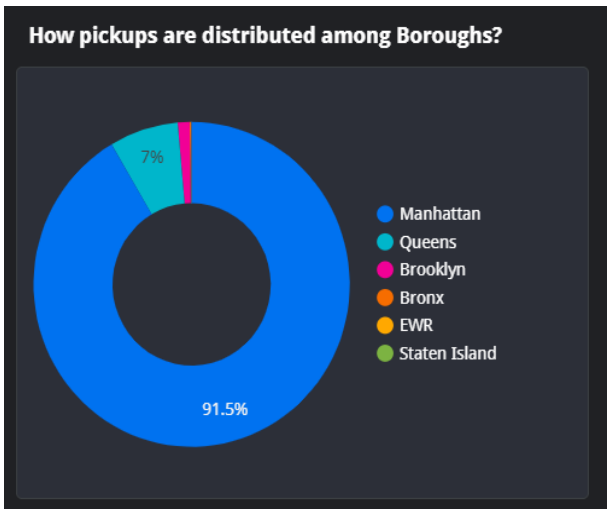


Figure 70: Predicted Pickups and there Distribution Monthly

But to update it for the live feed it may take 44 seconds to 1 minute as it fed into the Big Table.

The user can also find some useful comparison results using the month filter. For example: Let us consider we want to estimate the business in April 2020 and for that, we can find the comparison of the year 2018 and 2019 result which can be seen below:

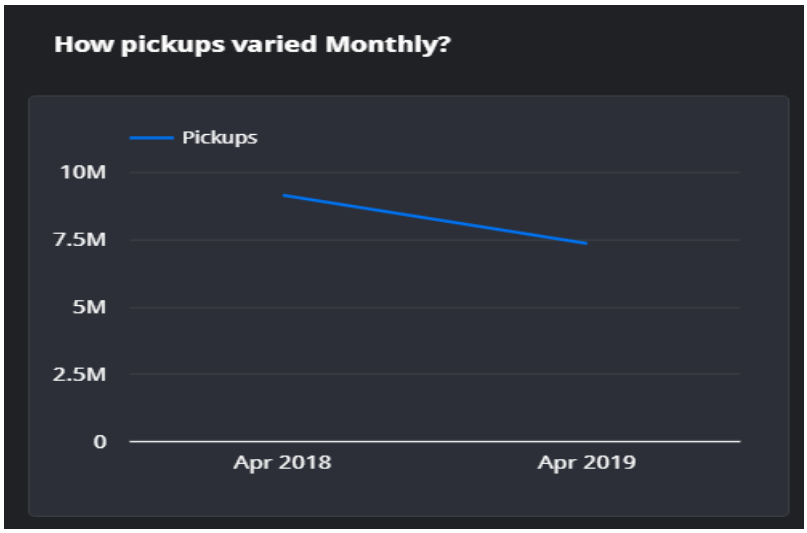


Figure 71: How pickup Vary yearly over the same month

Some other statistics with the details of zone id so that the user can have a better understanding.

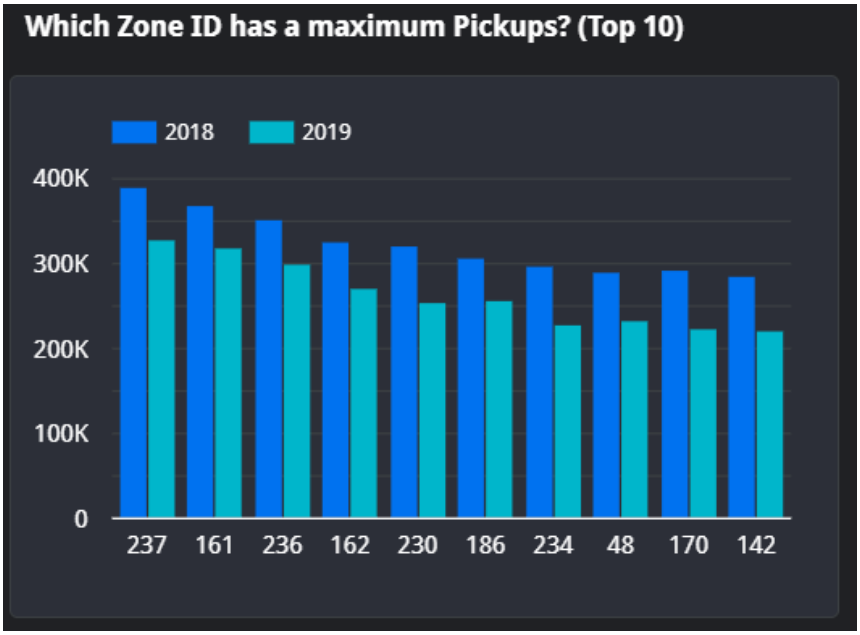


Figure 72: Hover on Bar to view the exact number

Further details can be checked is which days of the week in April can be the busiest for the Yellow Cab service and that can be found as Mondays and followed by Tuesdays which means people use more cabs on this week of days compare to than of the other days.

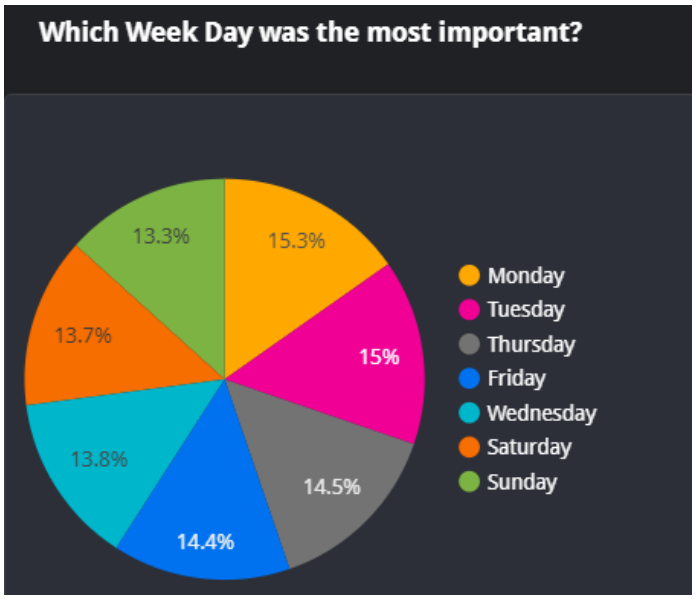


Figure 73: Variations of the number of pickups by Week Days

Development and Operational:

DevOps is used to deploy the code to production faster in an automated and repeatable manner. This helps the team and saves a large amount of time by deploying the code manually which can have human errors. But the use of development and operation allows agile development and provides better predictability, maintainability, reproducibility, cost efficiency and reduced risk and so on. The Lifecycle of DevOps work as follows:

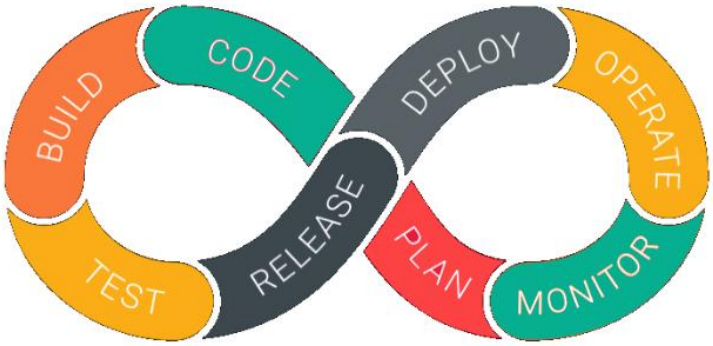


Figure 74: Development and Operationalize

Google App Engine:

It offers PaaS(platform as a service) that can help to host the development of the website and run across multiple servers. Hosted applications can run on their own on secure, reliable and which is independent of the location of any server and OS (Operating System).

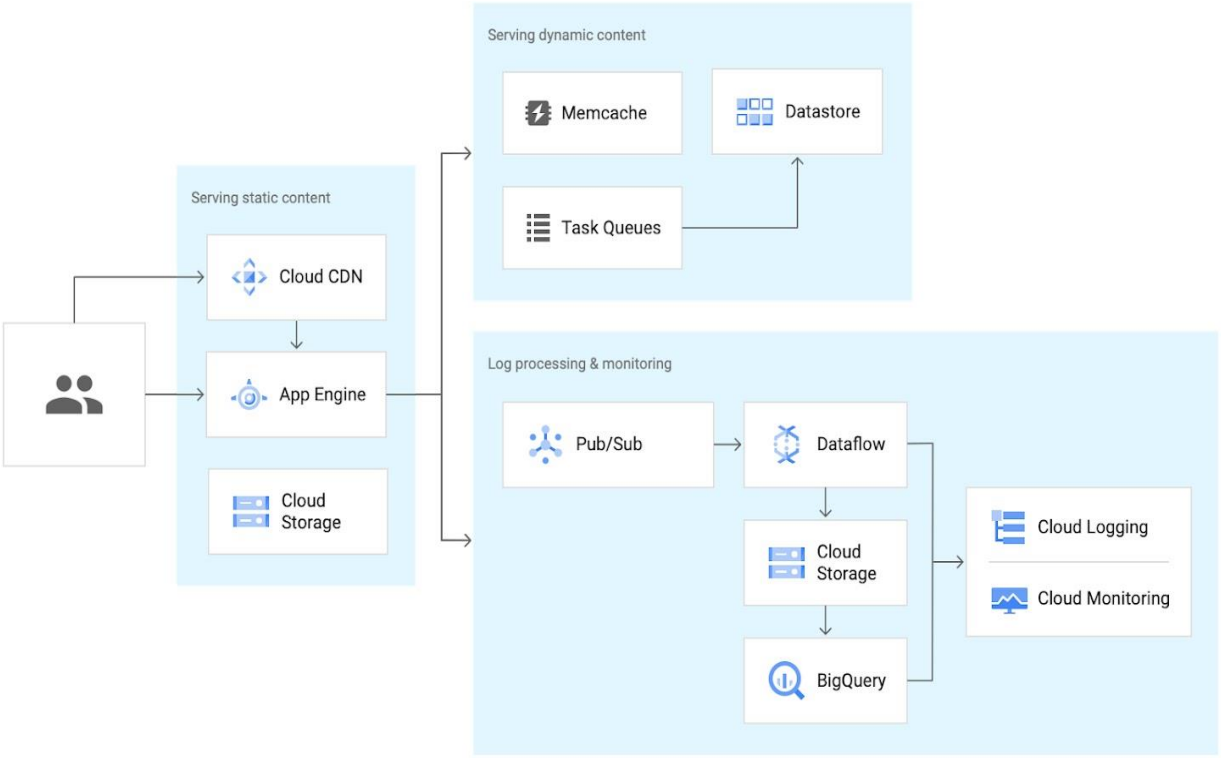


Figure 75: App Engine Architecture [4]

The model has been deployed on the Google Cloud Platform. The model was deployed on the development server initially then later I switch to the production server. I have a copy running in the development server on Google Cloud and the google cloud offers the scalability which helps to add extra nodes to the project at any time in the future. Hence the model can be scaled further if certain more conditions can be found.

Google also offers support of AutoMl which can also be used in the future to build a more accurate model. I also did A/B testing before deciding my interface. I did contact two of my friends who took Human-Computer Communication at U of R and I did A/B testing to select a suitable interface for the website.

Further Steps:

The work does not end with the operationalize phase:

• The model needs to be monitored refined.

• New attributes can be considered

• The delivery mechanism can be simplified with self-service reporting.

7. Tools:

The tools I plan to use are as follows:

- **Power BI:** To developed interactive visualizations and for data exploration.
- **Big Query:** Big Query is highly scalable and cost-effective to turn the raw data into informed business decisions.
- **Google Cloud Storage:** To store the entire data of around 30GB on the cloud.
- **Apache Spark:** Apache Spark is an open-source framework for a distributed cluster-computing and general purpose.
- **Plotly Python:** Plotly is an open-source browser-based graphing library for python which includes a declarative charting library, scientific charts, and statistical charts. **Plotly** is a company that makes visualization tools including a **Python** API library and also makes Dash and make use of the framework for building interactive web-based applications in python. Plotly by default makes visualizations and publishing online easy.
- **Jupyter iPython Notebook**: The notebook gives the ability to present the code in the form of a story that can be combined with rich texts, images,etc.
- **App engine to deploy the flask app**: App engine offers support to build scalable nodes and easily deploy the website on the cloud.
- **Flask to make the website:** The machine learning model can be easily deployed using Flask app.
- **Google Data Studio:** It is very similar to Power BI which will help to design interactive dashboards and customizable reports and it can be easily connected to Big Query, spreadsheets, analytics, and Google Ads. It also helps to visualize Big data when Power BI and TABLEAU fails.

8. Timeline of the Project work:

The Gantt chart for my project is as follows:

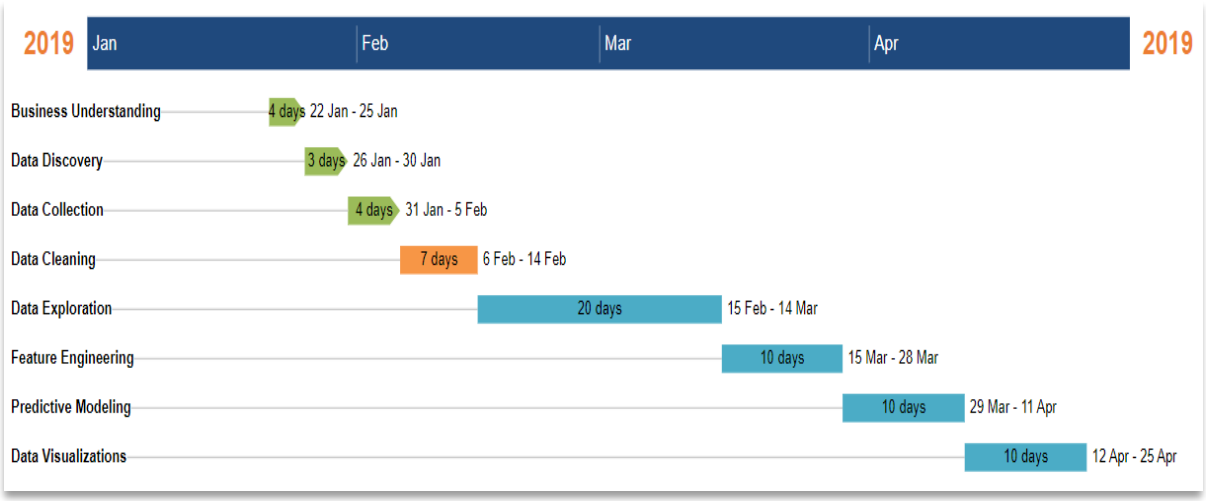


Figure 76: TimeLine Followed

9. Outcomes Obtained:

The outcome of the project will determine the number of pickups for a particular area at a particular time for Yellow taxi in New York City. To determine the number rides the user will have to select a location by drag and drop with the pointer marker shown on the map of New York.

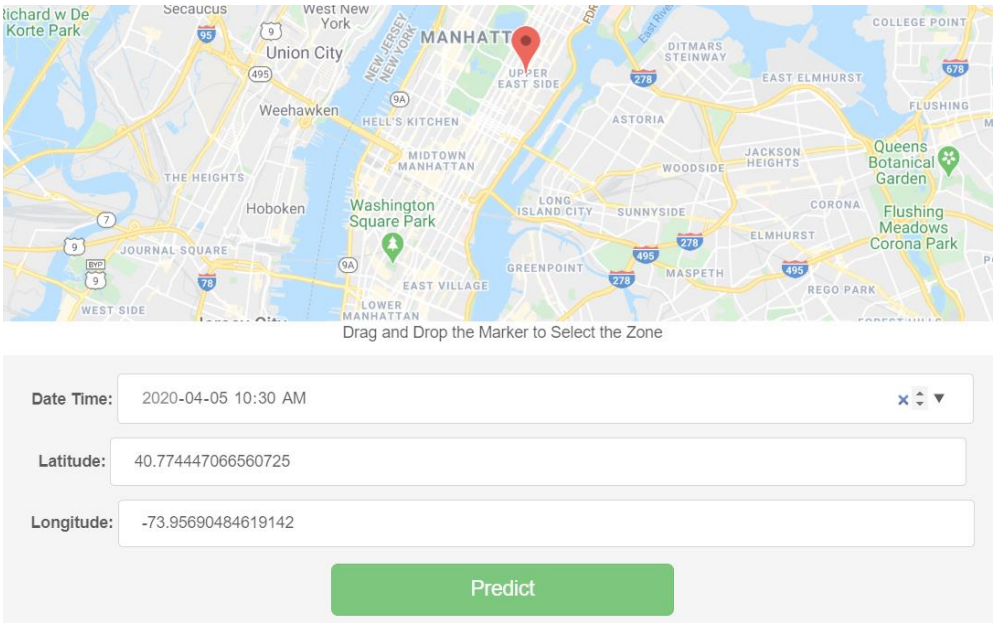


Figure 77: Website Home page

Once the user has selected the location on the map the longitude and the latitude of the location will be automatically entered. Also, the driver/user has to select the date (YYYY-MM-DD) and time (hr-min-AM/PM) in a particular format and then the user has to press the predict option which will then take to another page where the results with the number of pickups prediction will be shown along with the other details. This is shown in the below screenshot for the reference.

After this step, the user will be able to check the number of pickups for a particular location and a day which can be seen in the following way: Along with these, some other detail information can be found which might help the user to determine the accurate result such as Zone name, Borough name, time and temperature of the current day and forecasted weather for the upcoming week can also be seen on the dashboard. Also, the live weather of the zone will be scrapped from Google and will be taken into account for calculating the demanding areas around New York City. These scrapping of temperature may take up to 1 minute to load these reports.

The below figure shows the location mark on the map of the zone along with the zone name displayed on the dialog box with the number of pickups at the mentioned time and date.

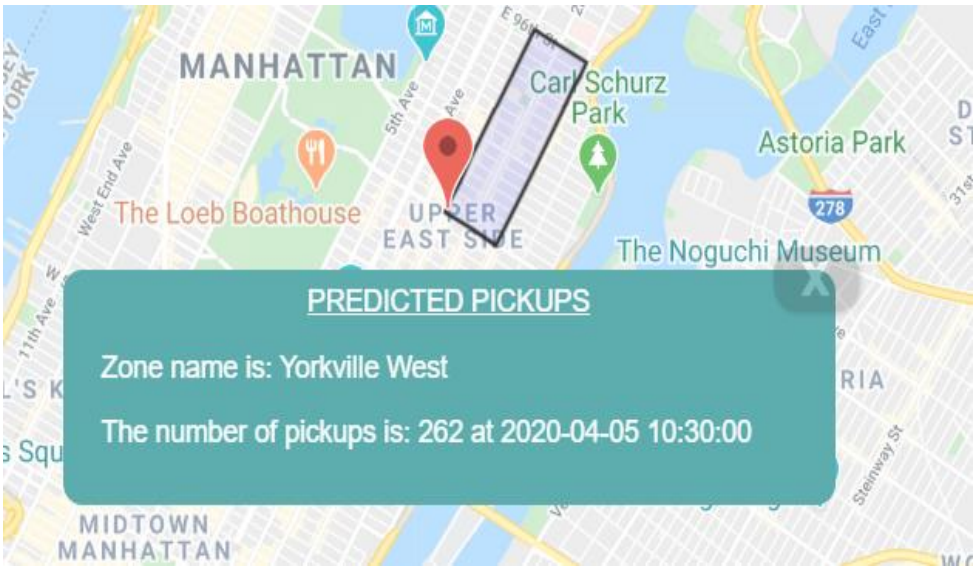


Figure 78: Predicted Pickups

The below figure shows the in-depth details of the location selected by the user such as zone id, borough name, the number of predicted pickups and the current time in New York City. It is the pickups that have taken one hour bracket into account.

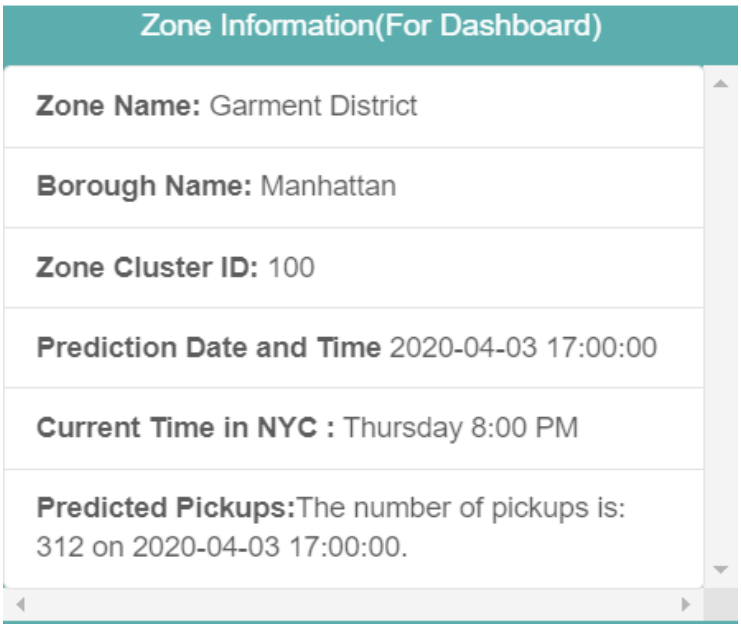


Figure 79: Zone Information Predicted

The side of the details will appear on the website which will provide the user about the next seven days of estimated temperature for the user's more personalized feel and better planning of the business.

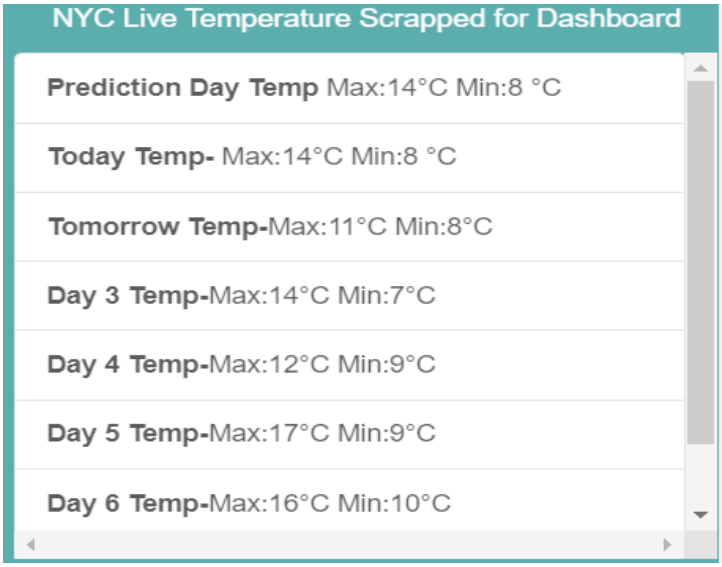


Figure 80: Live Temperature Scrapped

To move forward, to check some important statistics which are more useful for executives and also for the taxi drivers as the reports represent stats including high demand zone, high demand

time of the day including some weekdays stats where there will be more rush and demand for the cab services which will take the live dataset created and the live scrapping of temperature into the form of a dashboard which might take 1-2 minute time to load weekly reports. For this the user will have three options on the website which are 1) Predict, 2) Visualize forecasted Weekly Dashboard, 3) Visualize 2018-2019 Dashboard and can be seen as below:

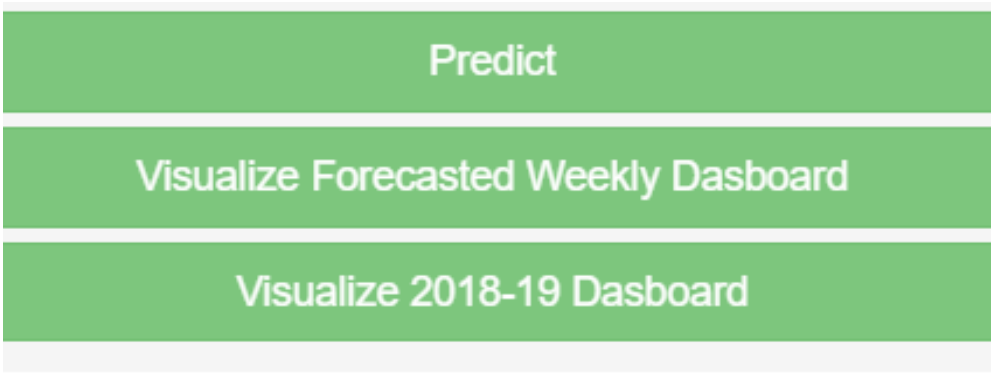


Figure 81: Three options for the User on the website

The predict options will help the user to check the number of pickups in any zone at any time. The second option is to visualize the weekly forecast where the user will be able to see the data for the upcoming week from the time he has selected including the weather live reports from google by scraping. Also, the user will have the chance to check out the comparisons between the reports of the year 2018 and 2019.

10. Originality and Novelty of proposal:

I have not found anyone exploring the TLC yellow data set from 2018 incorporating live data weather and publishing a fully functional website/app that can help TLC yellow taxi business grow. So this is my attempt to explore the dataset and get some valuable insights that might help the new York city TLC yellow cab taxi to grow their business in 2020. Apart from this, this will help the taxi drivers know the number of pickups they can expect in the zone they are currently operating in.

11. Potential number of users

The website made by me can be used by the NYC TLC agency to manage the rides of yellow

taxis in New York City. Every taxi driver can use this website to know the high demanding area requiring the most pickups and can move up to that place. NYC government has mentioned that there are **13,587 taxis** in New York City in the yellow cab introduction manual available on the NYC government site. [1] Hence the potential number of users is certainly **greater than 10,000**.

12. Business Rank of the users of the application

NYC yellow cab drivers are sinking in debt as per the article published in the Guardian.[3] One of the causes are they are finding difficulty in getting the customer and most of the time they spent idle just waiting for the customers. This is dropping the TLC yellow taxi business value. And due to this, the salary of the taxi drivers are dropping over the years. This will help to improvise their business plan and helps to increase profit. This will help to provide a better understanding of the taxi drivers as which area in New York City requires more number of taxis and has more amount of passengers.

13. Potential Business Value of Application:

TLC CAN use this website to enhance their business by sending the yellow cab at the right places at the right time. Now with the working website and the dashboards the company can easily enhance its revenue by sending the correct number of taxis at the demanding hotspots around the NYC city. Hence this web application can be ideal for Yellow taxi cab drivers and the business as well.

14. The difficulty of the Problem

This problem allows me to deal with Big Data as the ram of 25 GB was not enough while training the model and thus was breaking my traditional laptop system. This project covers the following:

- Fully developed and operational cloud-based published app or website.
- Also, this will help the TLC taxi agency to ensure that they are not having a shortage of taxis and they are sending the right number of taxis to correct location to meet the demand of the pickups at that location.
- Scrapping the live weather from google for the demand prediction.

- Not enough memory error when loading the data on pandas dataframe makes this problem a Big Data problem. This made me switch to PySpark with 2 clusters (1 master and 1 slave). Also, I took the help of Bigquery to explore Big Data for my case at a fast speed.

15. Prototype (Assessed code/solution environments)

The entire code for my project can be found on https://github.com/ssrbazpur/Envisioning-Yellow-Taxi-High-Demand-Areas-in-NYC-city

Solution Environment Setup:

- **Google Cloud:** Stored entire data on Google Cloud and make use of Big Query to retrieve the data.

Figure 82: Flask Deployment

NYC taxi contains all the data I require to build the project and Deploy Flask will contain my web app. Big Query on Google Cloud Platform helps to analyze Big Data data using Query.

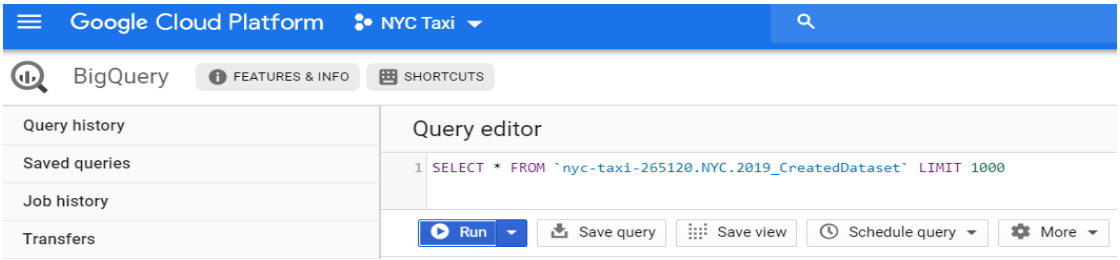


Figure 83: Big Query Editor

- **Pyspark Installation in Google Colab:**

I also installed Pyspark to get the support of RDD as pandas were not able to process such a large dataset.

https://github.com/ssrbazpur/Data-Science/blob/master/PySpark_Installation.ipynb

- **Deploy app on Google Cloud/AWS:** I deployed my model using Google App Engine:

https://deploy-flask-266818.appspot.com Google cloud

The below figure illustrates the analyzing of the machine learning model:

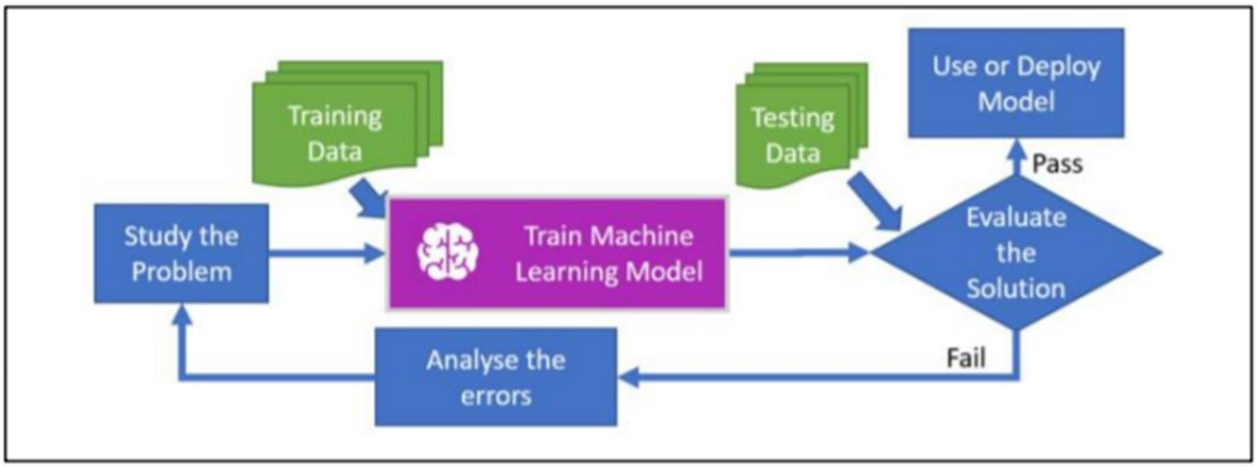


Figure 84: Analyzing Machine Learning Model

Initial Prototype that I proposed:

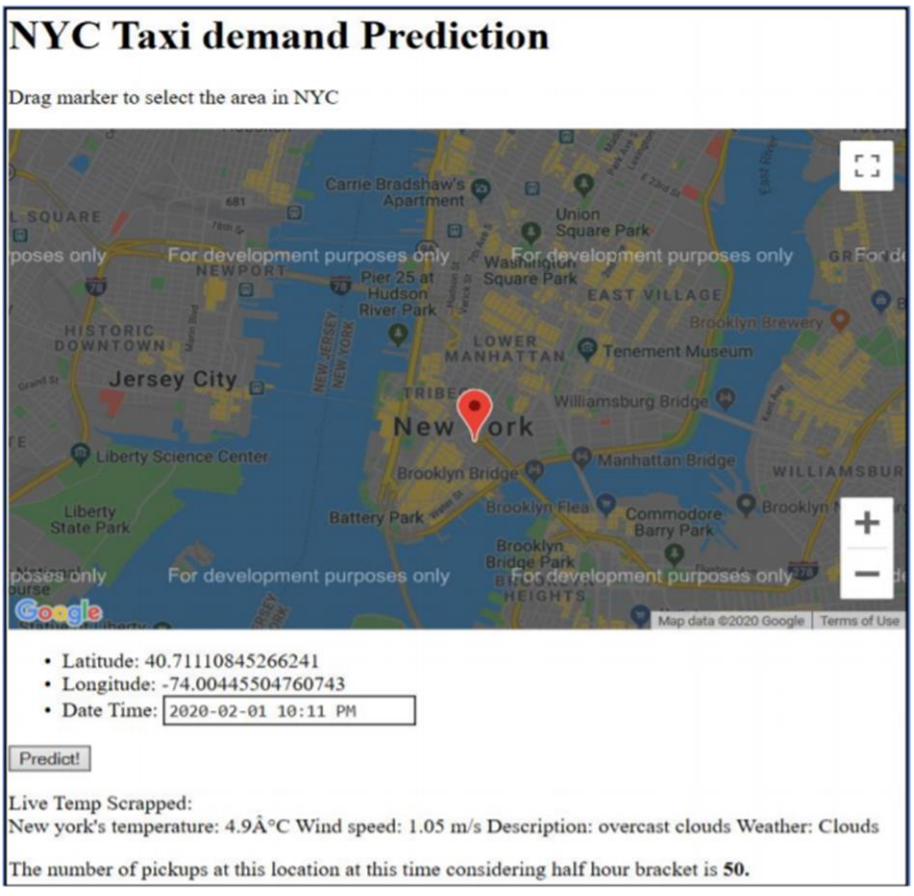


Figure 85: Proposed Prototype

Working Prototype:

The final developed prototype can be found at https://deploy-flask-266818.appspot.com/

As indicated in my initial prototype proposal section the final working prototype also incorporated the fully functional Dashboard and can be used by executives and businesses to take an effective decision that can help to increase the net revenue generated by the yellow taxi cabs in the NUC city.

16. Summarized Details:

So to summarize the project I would like to highlight a few of the following details. The project has covered the categories as below:

- Fully developed and operational cloud-based published app or website.
- Also, this will help the TLC taxi agency to ensure that they are not having a shortage of taxis and they are sending the right number of taxis to correct location to meet the demand of the pickups at that location.
- Scrapping the live weather from google for the demand prediction.
- Not enough memory error when loading the data on my laptop was solved using PySaprk that uses RDD to process the dataset.

Some other important cases that the project will look:

- Taxi drivers can know the right time to leave for their job.
- If due to weather the demand is too low some taxi drivers can take a day off and return to work when the demand for passenger pickup increases.Also,The taxi demand in a different location can be estimated using this idea.
- Taxi demand can be forecasted using a particular hour of the day, the weather of the day or day of the week and many more features that are yet to be explored.
- Multiple locations can have high demand hence this idea will help the TLC agency to redistribute yellow cab drivers around the city. Also, this can be used by TLC to make sure that the number of taxis is certain to meet the demand of the pickups required in that locality.

17. References:

[1] https://www.ny1.com/nyc/all-boroughs/transit/2019/05/31/nyc-yellow-cab-drivers earnings-have-fallen-44-percent-since-2013-taxi-and-limousine-commission-tlc says?cid=share_clip

[2] https://www1.nyc.gov/site/tlc/businesses/yellow-cab.page

[3] https://www.theguardian.com/us-news/2017/oct/20/new-york-yellow-cab-taxi-medallion-value-cost

[4] M. Malawski, M. Kuźniar, P. Wójcik, and M. Bubak, "How to Use Google App Engine for Free Computing," in IEEE Internet Computing, vol. 17, no. 1, pp. 50-59, Jan.-Feb. 2013.