# Data Analysis and Visualization of Green Taxi Records in New York

Sri Harsha Korukonda Bhattar
Cullent College of Engineering
University of Houston
Houston, USA
skorukondabhattar@uh.edu

Ankith Kandala
College of Natural Sciences and Mathematics
University of Houston
Houston, USA
akandala@uh.edu

Aashish Joshi
College of Natural Sciences and Mathematics
University of Houston
Houston, USA
ajoshi12@uh.edu

***Abstract*—Big cities like New York has one of the highest usage of taxis. Traditionally the data captured from the NYC Taxi & Limousine Commission was physically analyzed by various analysts, to determine which step would aid the people commuting via taxis. Due to exponential increase in taxi services, the data captured by NYC was in GB's, which was very difficult to analyze manually. There were millions of taxi rides per year in New York City. To overcome these hitches, Machine Learning and Cloud Computing were used to analyze such colossal dataset. Machine Learning can effortlessly analyze hundreds of GB within a fraction of seconds and expedite the process with the help of Cloud Computing. In this report, we will be using Machine Learning algorithms on AWS to make analysis and predictions on Green Taxi in New York City. This information can be used by numerous authorities and industries for their own purpose.**

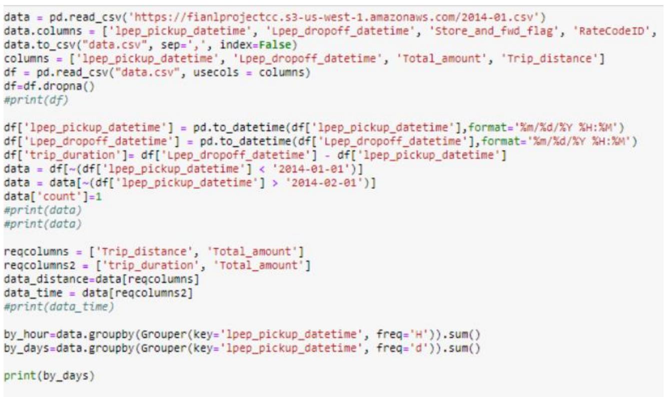
## I. Introduction

Transportation has been proved as the most vital service in large cities. Diverse modes of transportation are accessible. In large cities like New York, taxi mode of conveyance plays a foremost role and used as the best substitute for the general public use of transportation to get their necessities. For instance, by today in New York there are nearly 80,000 vehicles and 200,000 drivers in NYC Taxi and Limousine Commission. In order to provide very good taxi service and plan for effective integration in city transportation system, it is very important to analyze the demand for taxi transportation. The dataset provides the relating information such as the fare price, distance and Global Positioning System (GPS) data type for every taxi trip, including the time and location (latitude and longitude) of the pickup and drop-off. Our project is to leverage different AWS cloud services and build a machine learning platform which will analyze given data, find patterns and trends in it and then make future predictions based on user inputs and trained model. We will first process the raw data, analyze it, train the data and make training model and use it to make predictions of future dates along with customer interface in API Gateway.

## II. Description

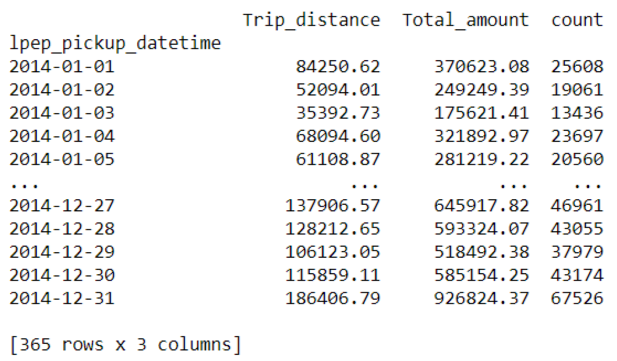
### A. *Preprocessing the Raw Data*

The very first step in making a good prediction is to search the source of the data. We are extremely thankful to TLC (Taxi and Limousine Commission) for providing us with the raw dataset for 6 years (2014 – 2019) for training our Machine Learning Models and predict the future patterns and trends. The dataset that we have received was of a raw data i.e., the column names and the datatypes of the content were different for different years of data. Hence, pre-processing of the data was a basic and necessary step in achieving our goal. We had to add relevant datatypes, remove the irrelevant columns which were independent to the data prediction. We chose Python as our programming language to filter the data and convert it into appropriate format. First step was to combine the data for the first five years (2014 - 2018) as make it as a training set. One of the major challenges that we have faced here was, we could not have combined the data set using Microsoft Excel. As the dataset were divided on a monthly basis, we had 60 files of data to be combined. Also, every dataset file had a minimum 0f 1.5 million records which was not possible to open with Excel. Hence, data pre-processing was done in Amazon Sage maker using Jupyter Notebook instance. For this step, we have used Sage Maker to combine the training dataset into a single file with only one header row. After that, we have eliminated the irrelevant columns such as "Tip Amount" and "Total tax" and have added some useful columns such as "Total Amount" and "Trip Duration" using the existing data. For our analysis and predictions, we have created two files of our data. One file has processed data and other file has data grouped in on hourly basis. Our code can be easily altered to fine tune the grouping even to seconds. We have added a new column called count in the data set. When the data is group, we will get sum of count values in the data which can be used to analyze customer behavior.



```
data = pd.read_csv('https://fianlprojectcc.s3-us-west-1.amazonaws.com/2014-01.csv')
data.columns = ['lpep_pickup_datetime', 'Lpep_dropoff_datetime', 'Store_and_fwd_flag', 'RateCodeID',
data.to_csv("data.csv", sep=',', index=False)
columns = ['lpep_pickup_datetime', 'Lpep_dropoff_datetime', 'Total_amount', 'Trip_distance']
df = pd.read_csv("data.csv", usecols = columns)
df=df.dropna()
#print(df)

df['lpep_pickup_datetime'] = pd.to_datetime(df['lpep_pickup_datetime'],format='%m/%d/%Y %H:%M')
df['Lpep_dropoff_datetime'] = pd.to_datetime(df['Lpep_dropoff_datetime'],format='%m/%d/%Y %H:%M')
df['trip_duration']= df['Lpep_dropoff_datetime'] - df['lpep_pickup_datetime']
data = df[~(df['lpep_pickup_datetime'] < '2014-01-01')]
data = data[~(df['lpep_pickup_datetime'] > '2014-02-01')]
data['count']=1
#print(data)
#print(data)

reqcolumns = ['Trip_distance', 'Total_amount']
reqcolumns2 = ['trip_duration', 'Total_amount']
data_distance=data[reqcolumns]
data_time = data[reqcolumns2]
#print(data_time)

by_hour=data.groupby(Grouper(key='lpep_pickup_datetime', freq='H')).sum()
by_days=data.groupby(Grouper(key='lpep_pickup_datetime', freq='d')).sum()

print(by_days)
```

This was part of the code used for preprocessing. In the data, one inconsistency we have observed was incorrect inputs. For example, in the data set of 2014, we have inputs of 2012, 2008 etc. So, after reading the data, we first remove unnecessary columns and then incorrect data rows. Some rows have incorrect data types. For example, we had rows where alphabets are there in the amount column which should have float values.



```
                    Trip_distance  Total_amount  count
lpep_pickup_datetime
2014-01-01               84250.62     370623.08  25608
2014-01-02               52094.01     249249.39  19061
2014-01-03               35392.73     175621.41  13436
2014-01-04               68094.60     321892.97  23697
2014-01-05               61108.87     281219.22  20560
...                           ...           ...    ...
2014-12-27              137906.57     645917.82  46961
2014-12-28              128212.65     593324.07  43055
2014-12-29              106123.05     518492.38  37979
2014-12-30              115859.11     585154.25  43174
2014-12-31              186406.79     926824.37  67526

[365 rows x 3 columns]
```

Here we can see data of 2014 that was grouped with groupby() function and later we have used sum() to get summated information.

## B. *Analysis of data to get behavioral pattern.*

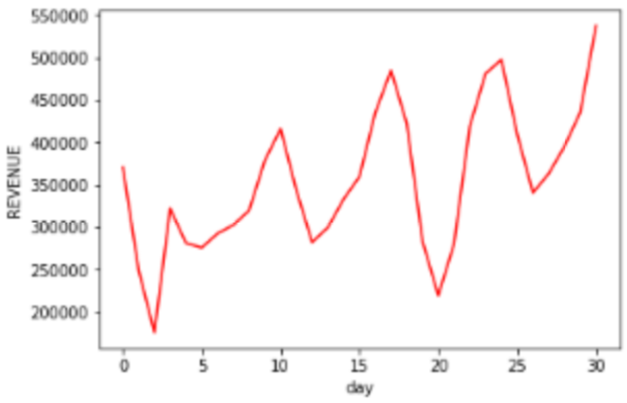
We have analyzed the data to show representations of how customer base is changing over time and what are the factors that are affecting it. Similar analysis was made to track changes in revenue. We were able to get of how traffic is changing on weekends and what are peak hours of traffic on a given day. Also, we have representations on how customer base is changing based on holidays, seasons.

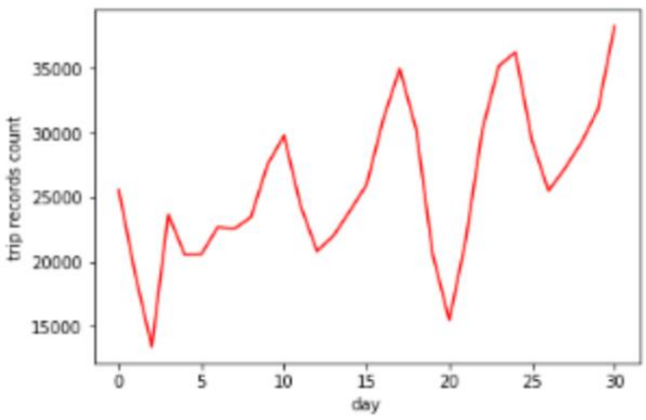
We were planning to include location data into our analysis to provide traffic analysis based on specific location, but we were unable to implement that, we see that as our future plan for the project. Another future plan would be to include weather reports on daily basis to the data and analyze it in coordination with this data. We have created a analysis system which could be easily used for any recurring data with very little changes.

Here we can see sample representation of how revenue and number of trips are changing on average month.

We can clearly see that number of trips is directly proportional to revenue. However, there were times were even with comparatively low trip count revenue was high pointing out at changes in surge pricing.

We have made certain observations like observed that Tuesday and peaks at weekends. Similar observations were made in yearly reports.





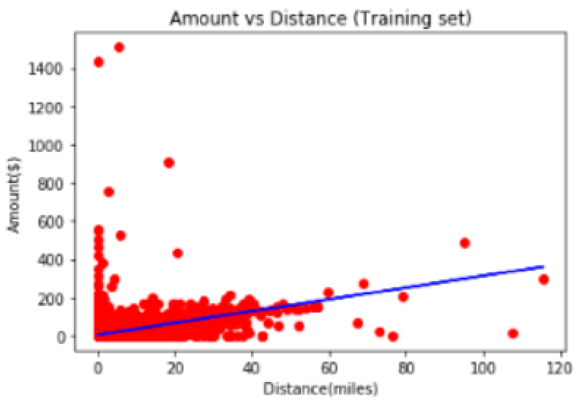
## C. *Choosing appropriate Machine Learning Algorithm*

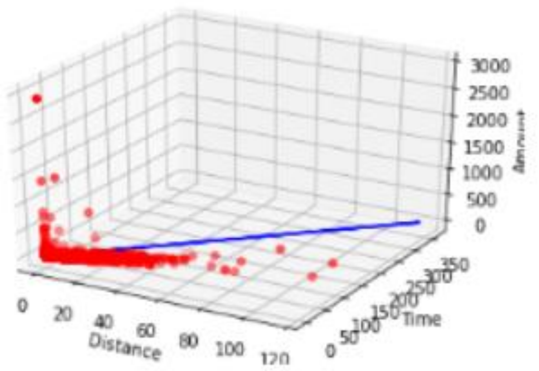
There are various machine learning algorithms. We had to choose one which has good accuracy on predicting the trends and fast enough to process all the data without much burden. We tested our preprocessed data with Multiple Linear Regression, Random Forest and Polynomial Regression. But we found that Multiple Linear Regression algorithm to be most efficient without losing much accuracy for this data.

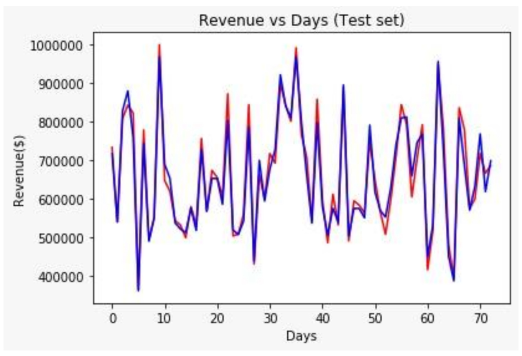
## D. *Predicition of future data.*

As mentioned earlier, we have taken 5 years data as training set. This data has about 70 million rows. We have given this as feed for training. We planned on making predictions for the year 2019 with these data models. We already have data for the year of 2019 from TLC's database. We used this actual data and compared these to the predicted values. We have observed that our predictions accuracy was Accuracy for trip pricing – 92.4% and Prediction Accuracy for revenue and customer base – 86.8%.

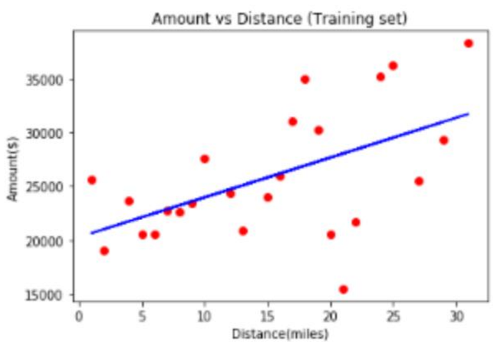
We have made two graphs to get comparison on how amount is changing with distance and how it is changing with time of travel. Then we made a multiple linear regression to check how fare is changing with both these parameters in consideration.







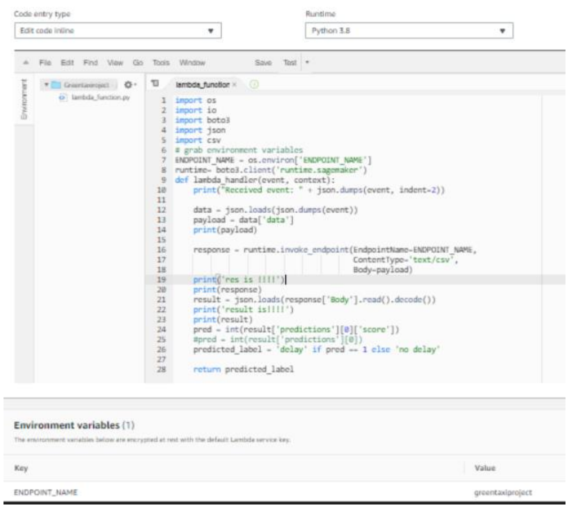
We have made representations of our predictions and actual data. Blue is actual data and Red line indicates predicted data. We see that our predictions were very close to actual readings.



## E. *Lambda function.*

Once we have the machine learning algorithms and predictions code in place, we have created a Sage maker endpoint which would reference this training model saved in Amazon S3. We have created a lambda function that would be triggered by an API gateway interface. Our aim was to make a serverless environment of our application.

Dedicated servers are helpful for business entities with predictable and steady customer base. However if the customer base is varying widely, it would be better to have serverless environment. It helps in shifting the responsibilities of maintenance to AWS



```python
import os
import io
import boto3
import json
import csv
# grab environment variables
ENDPOINT_NAME = os.environ['ENDPOINT_NAME']
runtime= boto3.client('runtime.sagemaker')
def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))

    data = json.loads(json.dumps(event))
    payload = data['data']
    print(payload)

    response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                       ContentType='text/csv',
                                       Body=payload)
    print('res is !!!!')
    print(response)
    result = json.loads(response['Body'].read().decode())
    print('result is!!!!')
    print(result)
    pred = int(result['predictions'][0]['score'])
    #pred = int(result['predictions'][0])
    predicted_label = 'delay' if pred == 1 else 'no delay'

    return predicted_label
```

**Environment variables** (1)

The environment variables below are encrypted at rest with the default Lambda service key.

| Key | Value |
|---|---|
| ENDPOINT_NAME | greentaxiproject |

Here we have created an environment variable and provided the name of our sage maker endpoint created in our Jupiter instance.

## F. *API Gateway.*

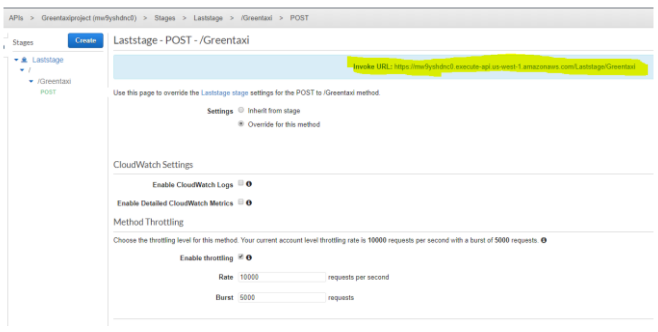
We have made two different types of data predictions. One set of predictions would provide customer centric data. For example, customers can use this to get estimate of trip fare based on individual parameters like trip distance and time of travel. Another set of predictions would be Industry centric. It would provide predictions to the green taxi management which based on historic data.

In the Customer centric API Gateway, customer can provide distance and time to get predicted trip fare.

In the Industry centric API Gateway, management can provide information like date to get predicted revenue and customer base for that date which would be 92% accurate. They can use this information to plan their resources better. For example, they can plan more resources for any upcoming holiday based on the predicted traffic.

Government and public works departments can used data from multiple sources of public transport to get a cumulative report of public behavior and plan strategies accordingly.

When ever a someone queries in the API Gateway, lambda function is triggered. It invokes Sage maker endpoint which has the training model. This is then return


APIs > Greentaxiproject (mw9yshdnc0) > Stages > Laststage > /Greentaxi > POST
Stages
Create
Laststage
/
/Greentaxi
POST

Laststage - POST - /Greentaxi

**Invoke URL:** https://mw9yshdnc0.execute-api.us-west-1.amazonaws.com/Laststage/Greentaxi

Use this page to override the Laststage stage settings for the POST to /Greentaxi method.

**Settings** ○ Inherit from stage [✓ ● Override for this method]

CloudWatch Settings

**Enable CloudWatch Logs** ☐

**Enable Detailed CloudWatch Metrics** ☐

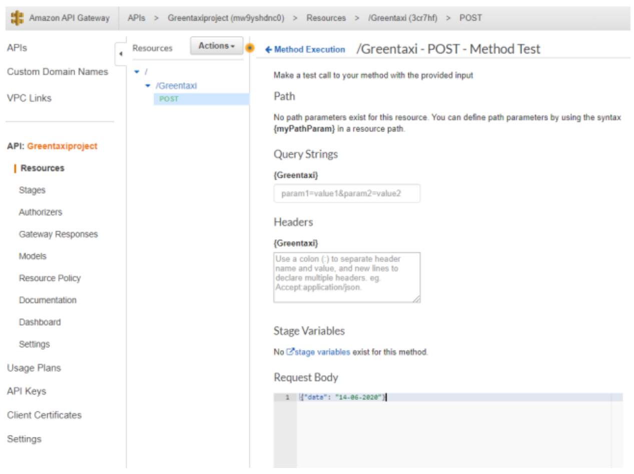
Method Throttling

Choose the throttling level for this method. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests.

**Enable throttling** ✓

**Rate** 10000 requests per second

**Burst** 5000 requests



Make a test call to your method with the provided input

No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.

Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.

No stage variables exist for this method.

```
{"data": "14-06-2020"}
```

The return value will be parsed by lambda again and It is sent back as prediction result to the end user in API gateway interface. We have tested our application by providing few dates in year 2020 and we could see outputs for our predictions with estimates of number of trips and estimate revenue.

## III. Challenges

We have encountered multiple challenges as we have progressed with the Project.

First challenge that we have faced was using our AWS educate account due to permission limitations for implementing various tools from AWS. Hence, we had to choose our personal AWS account to implement the project.

Next challenge was the data pre-processing. The data from every year had either a different format or a different column name. Also, there was a lot of redundant data which we had to get rid of. Apart from this, there were irregularities and lot of unnecessary data which was present. One such example that can be thought of is data of year 2008 and 2009 present in dataset of year 2015. There would be few missing data and also few files had noise added to them.

Third challenge that we have encountered was the number of columns of the dataset for each month. As we had the dataset in monthly format, the size of data set was around 200 MB per file, but the number of rows within each dataset was around 1.5 million which was impossible to be opened by Microsoft excel as it exceeded the maximum limit per sheet.

One of the major hurdles that was encountered by us was which machine learning algorithm should be implemented for such form of data. One solution that we sought was the research. After researching extensively, we narrowed downed the algorithms to Regression and Clustering.

We tried implementing k-means clustering algorithm for which we were unable to obtain expected output. As the algorithm divides the final data into binary output of 0's and 1's, and the data being to similar to each other, we were unable to get the conclusive proof of accuracy that needs to be obtained for the Algorithm to be called as a "well-trained " algorithm. Even after trying to predict the accuracy of the Algorithm using the accuracy prediction and using confusion matrix, we were unsatisfied by the output and hence we chose to move on with Multiple Linear Regression as our desired Machine Learning Algorithm.

Next challenge that we have faced was of integrating the location data (latitude and longitude) with our final predictions. As the dataset contained negative latitude values, we were a bit stuck on how to comprehend this data.

Also, we did not implement post-GRE SQL database as we intended to do initially. Main reason for omitting this is S3 was a simpler and efficient way to handle our dataset.

## III. FUTURE DEVELOPMENTS

We would like to develop GUI interface for the applications where end users would be allowed to provided multiple independent variables and get predictions.

We would like to integrate weather and location constraints to the data.

This application can be further developed to handled streaming data and training models being updated for accuracy as per latest data.

## IV. Roles

We have divided the roles equally among the teammates.

Sri Harsha has taken up Machine Learning Algorithm named "Polynomial Regression" and has worked on Data pre-processing.

Aashish Joshi has worked on "Multiple Linear Regression" which we have used in this report, along with creating the Lambda function and Endpoint.

Ankith Kandala has implemented on "Random Forest Regression" and "k-means Clustering". Also he has created API gateways for invoking Lambda Endpoint.

## Acknowledgment

We would like to thank the Taxi & Limousine Commission of New York City for the data capitalized in this paper and Prof. Weidong Shi for providing credit in our AWS educate.

## References

[1] NYC Taxi & Limousine Comission. : https://www1.nyc.gov/site/tlc/about/about-tlc.page

[2] Taxicab Factbook. https://www1.nyc.gov/site/tlc/about/fact-book.page

[3] https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html

[4] https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html

[5] https://docs.aws.amazon.com/step-functions/latest/dg/connect-sagemaker.html

[6] https://aws.amazon.com/blogs/machine-learning/call-an-amazon-sagemaker-model-endpoint-using-amazon-api-gateway-and-aws-lambda/