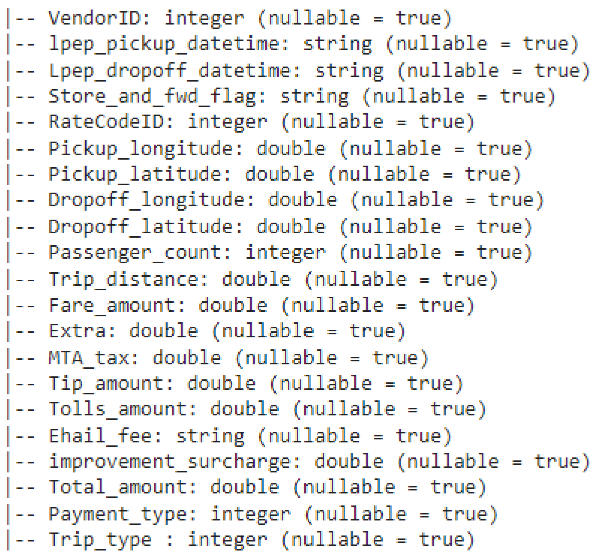
New York City Taxi Trips

Muhammad Umar Salman
`21010241@mbzuai.ac.ae`

1 Introduction

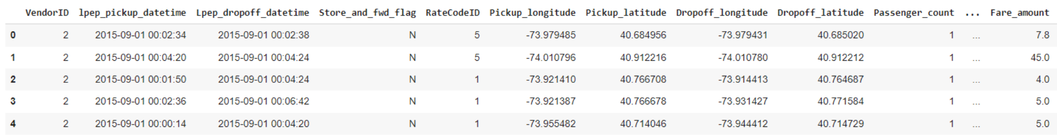
This project looks at the data concerning the trips made by New York City Taxi's and Limousine in specific the Green Taxis which as opposed to the yellow ones are taxis that are not allowed to pick up passengers inside the densely populated areas of Manhattan. We look at features such the average fare amounts and total amounts for certain trips and use clustering methods to see which areas are more popular for pickups and drop-offs. The data we will look at insights are collected over the month of September 2015 where the data can be found from [1]. There are a total of 1494926 rows in the dataset and 21 columns whose metadata information can be found from [2]. We use the pySpark which is a python wrapper for Spark which executes much faster by caching data in-memory across multiple parallel operations which much faster when working with data of this size. The columns can be visualized using Spark's print schema which will also describe the data types for each column or feature which can be seen here in Figure 1



```
|-- VendorID: integer (nullable = true)
|-- lpep_pickup_datetime: string (nullable = true)
|-- Lpep_dropoff_datetime: string (nullable = true)
|-- Store_and_fwd_flag: string (nullable = true)
|-- RateCodeID: integer (nullable = true)
|-- Pickup_longitude: double (nullable = true)
|-- Pickup_latitude: double (nullable = true)
|-- Dropoff_longitude: double (nullable = true)
|-- Dropoff_latitude: double (nullable = true)
|-- Passenger_count: integer (nullable = true)
|-- Trip_distance: double (nullable = true)
|-- Fare_amount: double (nullable = true)
|-- Extra: double (nullable = true)
|-- MTA_tax: double (nullable = true)
|-- Tip_amount: double (nullable = true)
|-- Tolls_amount: double (nullable = true)
|-- Ehail_fee: string (nullable = true)
|-- improvement_surcharge: double (nullable = true)
|-- Total_amount: double (nullable = true)
|-- Payment_type: integer (nullable = true)
|-- Trip_type : integer (nullable = true)
```

Figure 1: Schema of Columns

We are also attaching a snapshot of the first 5 rows of the data or 5 trips to show a clearer image of what the data looks like. This can be seen in Figure 2



	VendorID	lpep_pickup_datetime	Lpep_dropoff_datetime	Store_and_fwd_flag	RateCodeID	Pickup_longitude	Pickup_latitude	Dropoff_longitude	Dropoff_latitude	Passenger_count	...	Fare_amount
0	2	2015-09-01 00:02:34	2015-09-01 00:02:38	N	5	-73.979485	40.684956	-73.979431	40.685020	1	...	7.8
1	2	2015-09-01 00:04:20	2015-09-01 00:04:24	N	5	-74.010796	40.912216	-74.010780	40.912212	1	...	45.0
2	2	2015-09-01 00:01:50	2015-09-01 00:04:24	N	1	-73.921410	40.766708	-73.914413	40.764687	1	...	4.0
3	2	2015-09-01 00:02:36	2015-09-01 00:06:42	N	1	-73.921387	40.766678	-73.931427	40.771584	1	...	5.0
4	2	2015-09-01 00:00:14	2015-09-01 00:04:20	N	1	-73.955482	40.714046	-73.944412	40.714729	1	...	5.0

Figure 2: Snapshot of Data

2 Tasks

The project included 8 tasks where we had to use the pyspark framework along with algorithms we learnt in class to solve the following problems and to provide some extra insight as to how we coud improve the existing system.

2.1 Task 1

Report how many rows and columns the data file have?
The first task required us to download and import the data using Spark's read.csv functionality and to return the total number of rows and columns of the dataset. We simply used

```
df.count() and len(df.columns)
```

to extract that information as 1494926 and 21 respectively.

2.2 Task 2

Collect all the trip distances for all trips. Plot a histogram of the trip distances. You can play with the number of bins or use the default
In task 2 we had to plot a histogram of the trip distances for all trips. Since the distances was a continuous variable we used binning to get bins of distances that the green taxis travelled. Here we looked at the Trip distance columns and plotted a histogram using python's plotly library. Since we noticed that the max trip distance was 603.1 and most of the trip distances lied with under the range of 10 we changed our y axis such that it would should bin sizes of 1 from 0 to 20 and then one bin would show all the values > 20 which were 3394 (Not a large value when compared to the total number of trips). So to avoid a large y axis or all the starting trip distances to be binned into one bin we plotted our histogram as shown in Figure 3

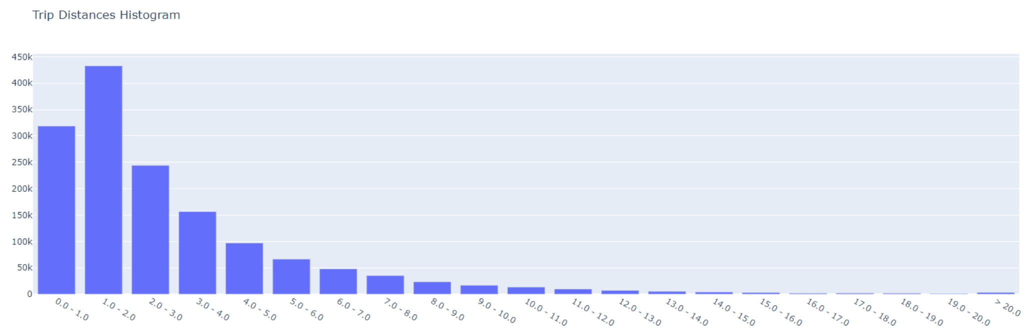


Figure 3: Histogram of Trip Distances

2.3 Task 3

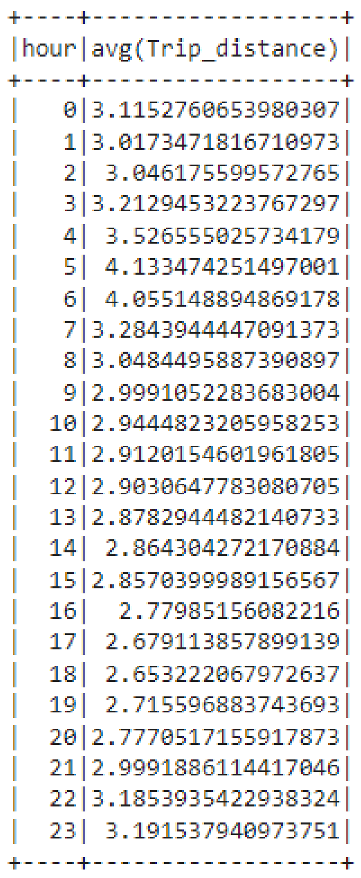
Report mean trip distance grouped by pick-up hour of day.
In the third task we had to take the mean trip distances based on the pick up hour of the day. To achieve this we first converted our pickup datetime column to a derived column which looked at the hour of the column. The line of code is shown below

```
df = df.withColumn("hour", hour("lpep_pickup_datetime"))
```

We then using the pick up hour values group by on that column while taking the average or mean of the trip distances which can be seen in the code below

```
df.groupBy("hour").mean("Trip_distance").orderBy('hour').show(25)
```

The output of the task is shown below in Figure 4



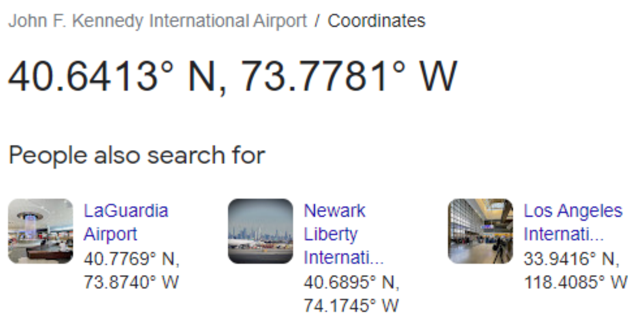
```
+----+------------------+
|hour|avg(Trip_distance)|
+----+------------------+
|   0|3.1152760653980307|
|   1|3.0173471816710973|
|   2| 3.046175599572765|
|   3|3.2129453223767297|
|   4| 3.526555025734179|
|   5| 4.133474251497001|
|   6| 4.055148894869178|
|   7|3.2843944447091373|
|   8|3.0484495887390897|
|   9|2.9991052283683004|
|  10|2.9444823205958253|
|  11|2.9120154601961805|
|  12|2.9030647783080705|
|  13|2.8782944482140733|
|  14| 2.864304272170884|
|  15|2.8570399989156567|
|  16|  2.77985156082216|
|  17| 2.679113857899139|
|  18| 2.653222067972637|
|  19| 2.715596883743693|
|  20|2.7770517155917873|
|  21|2.9991886114417046|
|  22|3.1853935422938324|
|  23| 3.191537940973751|
+----+------------------+
```

Figure 4: Mean Trip Distances by pickup hour

2.4 Task 4

We'd like to get a rough sense of identifying trips that terminate at one of the NYC area airports. Can you provide a count of how many transactions fit this criteria, the average fair, and any other interesting characteristics of these trips.

In this task we were required to choose an airport and find the number of trips that ended at this airport and the average fare of these trips to this airport. The airport we chose was JFK (John F. Kennedy International Airport). We can see the longitude and latitude values for JFK in Figure 5



John F. Kennedy International Airport / Coordinates

40.6413° N, 73.7781° W

People also search for

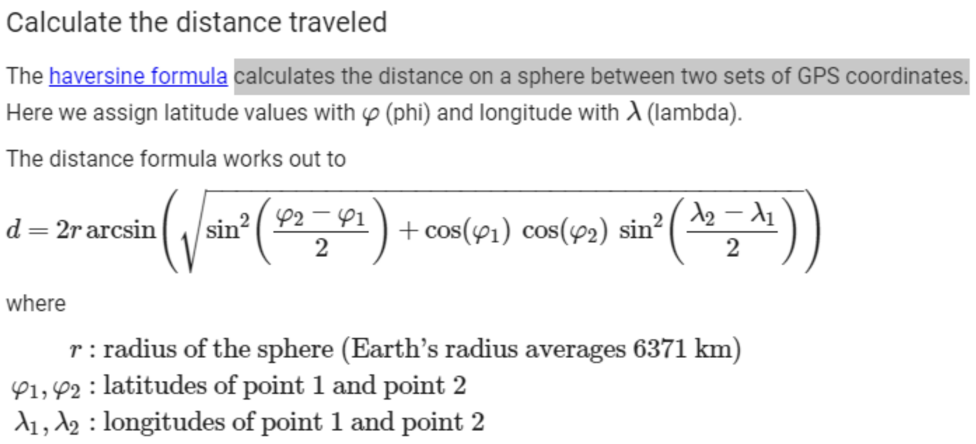
LaGuardia Airport
40.7769° N, 73.8740° W

Newark Liberty Internati...
40.6895° N, 74.1745° W

Los Angeles Internati...
33.9416° N, 118.4085° W

Figure 5: JFK Longitude and Latitude

In this section we use the Haversine Formula [1] seen in Figure 6,which calculates the distance on a sphere between two sets of GPS coordinates. We look at the drop-off longitude and latitude and calculate the Haversine distance from the JFK airport longitude and latitude and see which drop-off coordinates come under a 2 km radius. We classify those trips as trips to the JFK airport. The reason we have chosen 2 km as a radius is to encapsulate the error that might come from the geo-location coordinates.



Calculate the distance traveled

The haversine formula calculates the distance on a sphere between two sets of GPS coordinates. Here we assign latitude values with φ (phi) and longitude with λ (lambda).

The distance formula works out to

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\ \cos(\varphi_2)\ \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

where

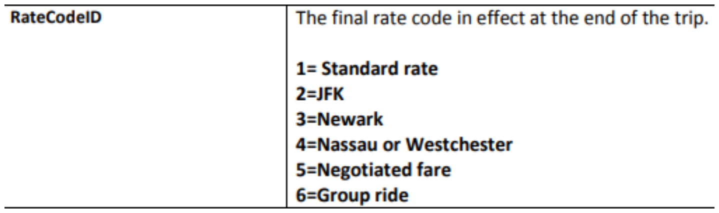
r : radius of the sphere (Earth's radius averages 6371 km)
φ_1, φ_2 : latitudes of point 1 and point 2
λ_1, λ_2 : longitudes of point 1 and point 2

Figure 6: Haversine Formula

Through filtering all the drop-off longitude and latitude which are under 2 km from the JFk cordinates we get a count of **12948** trips to the JFK airport.

```
temp.filter(col('DistanceToJFK') <= 2.0)
```

However something interesting that we found was that there is a column in the data which can be verified from [2] which is RateCodeID which identifies as the "The final rate code in effect at the end of the trip" where the RateCodeID 2 is the end trip code for JFK. A snapshot of the column metadata is shown in the Figure 7

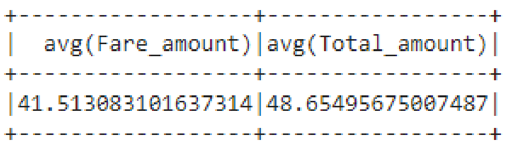


RateCodeID	The final rate code in effect at the end of the trip. **1= Standard rate** **2=JFK** **3=Newark** **4=Nassau or Westchester** **5=Negotiated fare** **6=Group ride**

Figure 7: RateCodeID Column description

This essentially means that all trips with this RateCodeID 2 should have a drop-off location at JFK. However while filtering the data on RateCodeID we see that only 4435 trips with that ID and then after further analysis we saw that out of all the trips with that ID only 2354 lied within a 2 km radius of the JFK coordinates.
Below Figure 8 shows the average fare amount and the average total amount for the 12948 trips to the JFK airport.

[1]https://en.wikipedia.org/wiki/Haversine_formula

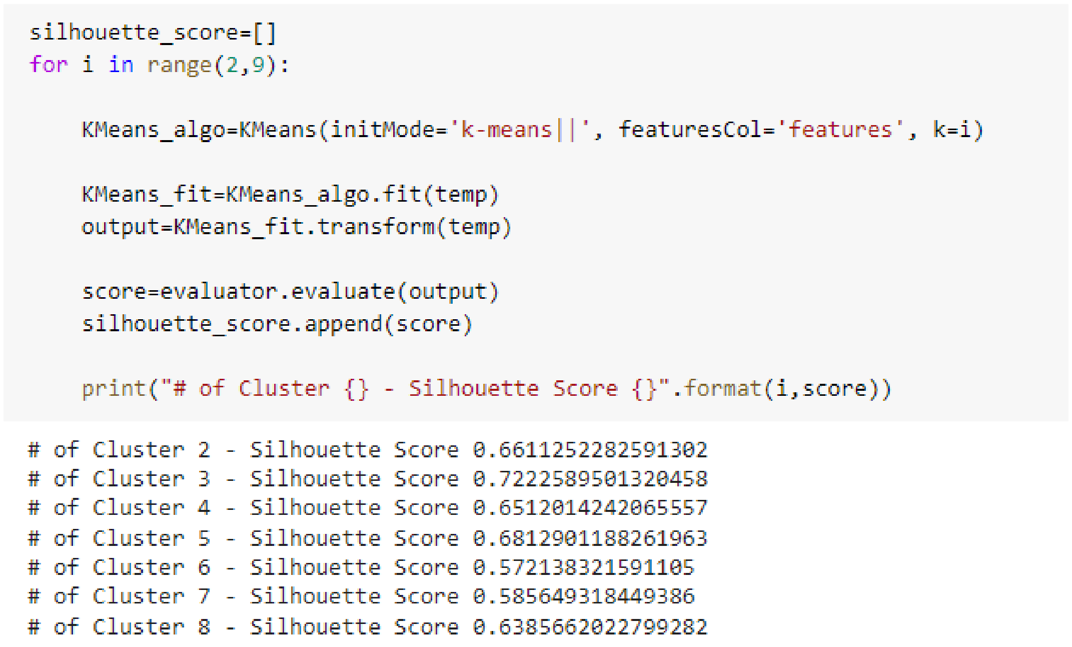


```
+------------------+-----------------+
|  avg(Fare_amount)|avg(Total_amount)|
+------------------+-----------------+
|41.513083101637314|48.65495675007487|
+------------------+-----------------+
```

Figure 8: Average Fare and Total Amount to JFK

2.5 Task 5

Do two clusterings on pick-up and drop-off locations respectively. Choose a proper number of clusters k and report the centroids. Is there much difference between centroids for pick-up and drop-off? Note: do not set the maximal k too large in your code. k from 2 to 10 - 20 should be enough to try.
In this task we aim to find the cluster centroids for the optimal number of clusters k for the drop off locations and the pickup locations. We run the clustering algorithm using Spark's MLib clustering implementation which can be seen in Figure 9



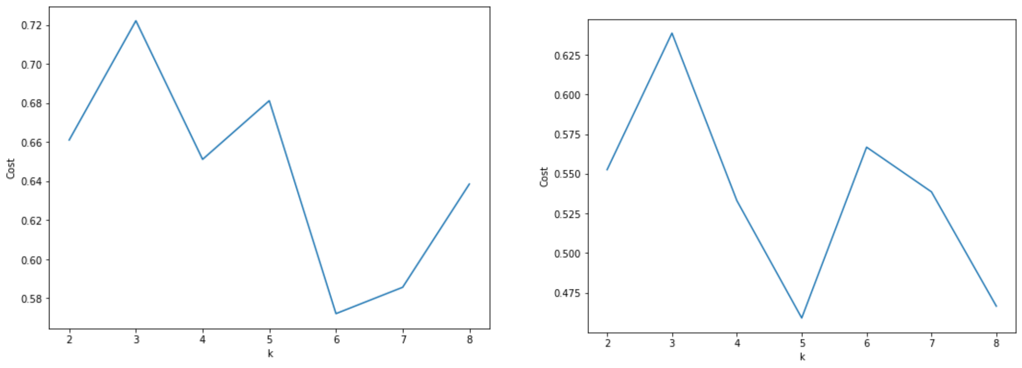
```
silhouette_score=[]
for i in range(2,9):

    KMeans_algo=KMeans(initMode='k-means||', featuresCol='features', k=i)

    KMeans_fit=KMeans_algo.fit(temp)
    output=KMeans_fit.transform(temp)

    score=evaluator.evaluate(output)
    silhouette_score.append(score)

    print("# of Cluster {} - Silhouette Score {}".format(i,score))
```

```
# of Cluster 2 - Silhouette Score 0.6611252282591302
# of Cluster 3 - Silhouette Score 0.7222589501320458
# of Cluster 4 - Silhouette Score 0.6512014242065557
# of Cluster 5 - Silhouette Score 0.6812901188261963
# of Cluster 6 - Silhouette Score 0.572138321591105
# of Cluster 7 - Silhouette Score 0.585649318449386
# of Cluster 8 - Silhouette Score 0.6385662022799282
```

Figure 9: Clustering Algorithm

Here we run both clustering algorithms on pickup and drop-off locations while look at their results for optimal k clusters. We use the **The Silhouette Method** where the silhouette value measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation). The range of the Silhouette value is between +1 and -1. A high value is desirable and indicates that the point is placed in the correct cluster. If many points have a negative Silhouette value, it may indicate that we have created too many or too few clusters [3].As mentioned before that a high Silhouette Score is desirable. The Silhouette Score reaches its global maximum at the optimal k. This should ideally appear as a peak in the Silhouette Value-versus-k plot. Thus we run K-means clustering on both drop-off and pickup locations while first getting the optimal k clusters through the Silhouette method mentioned above. Figure 10 show the Silhouette Value-versus-k plot for both coordinates.

Looking at both Silhouette Value-versus-k plots we can see that the maximum value for both lies at k=3, thus we chose k = 3 as our optimal k and run our K-means algorithm with k++ initialization to get our cluster centers which can be seen below. It can also be seen that the cluster centers are almost exactly similar to each other for both sets of coordinates in that they in most places at the 3rd



(a) Pickup (b) Dropoff

Figure 10: Silhouette Value-versus-k plot for pickup and dropoff coordinates.

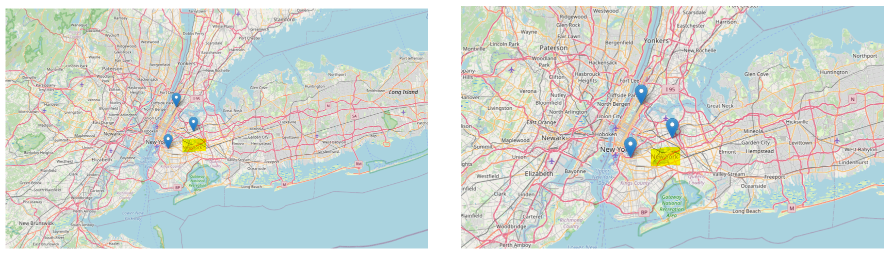
decimal place of the values. Plotting both centers on New York's map in Figure 13 also show's that both cluster centers follow are closely the same.

Pickup Centers

```
[-73.93712528  40.80841123]
[-73.966949    40.69311675]
[-73.87355798  40.74084974]
```

Dropoff Centers

```
[-73.93916513  40.80088071]
[-73.96846485  40.69569456]
[-73.85634645  40.73422424]
```



(a) Pickup (b) Dropoff

Figure 11: Cluster centers for both sets of coordinates on New York map.

2.6 Task 6

Cluster pick-up and drop-off locations. Choose a proper number of clusters k and report the centroids (you can use google maps to show centroids). What is the percentage of trips for each cluster? What is percentage of trips where pick-up and drop-off are in the same cluster? What have you learnt from this?
In this task we concatenate the longitude and latitude of dropoff and pickup coordinates together as a single set and run our K-means clustering algorithm on it. Again like discussed above we first find the optimal k clusters for the clustering which we do through the Silhouette method. This can be seen as to again be k=3 as shown in Figure 12. And it's centers on the New York map can be seen in the Figure 13 which closely follows the same pattern as the individual cluster centers. The values of the clusters are shown below and the predictions for each trip example can also be seen in Figure 14.

Concatenated (Pickup + Dropoff) Centers

```
[-73.96791234  40.69440503]
[-73.86269197  40.73616492]
[-73.93755649  40.80349395]
```

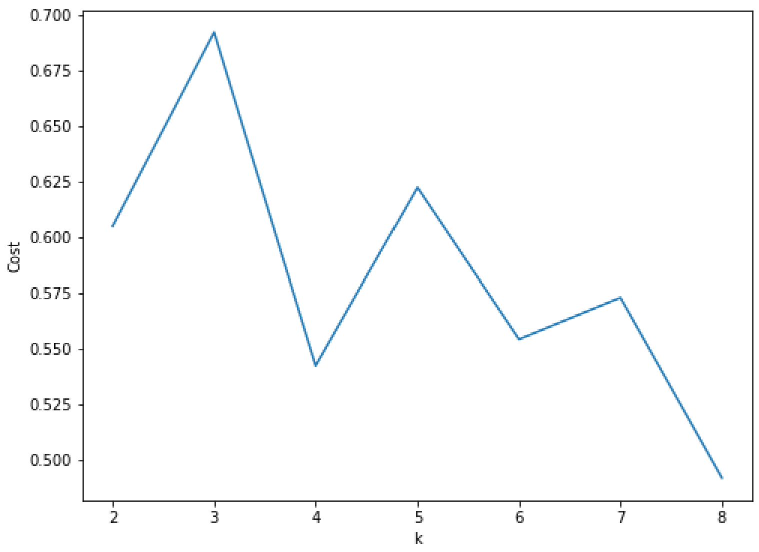


Figure 12: Silhouette Value-versus-k plot for concatenated coordinates.

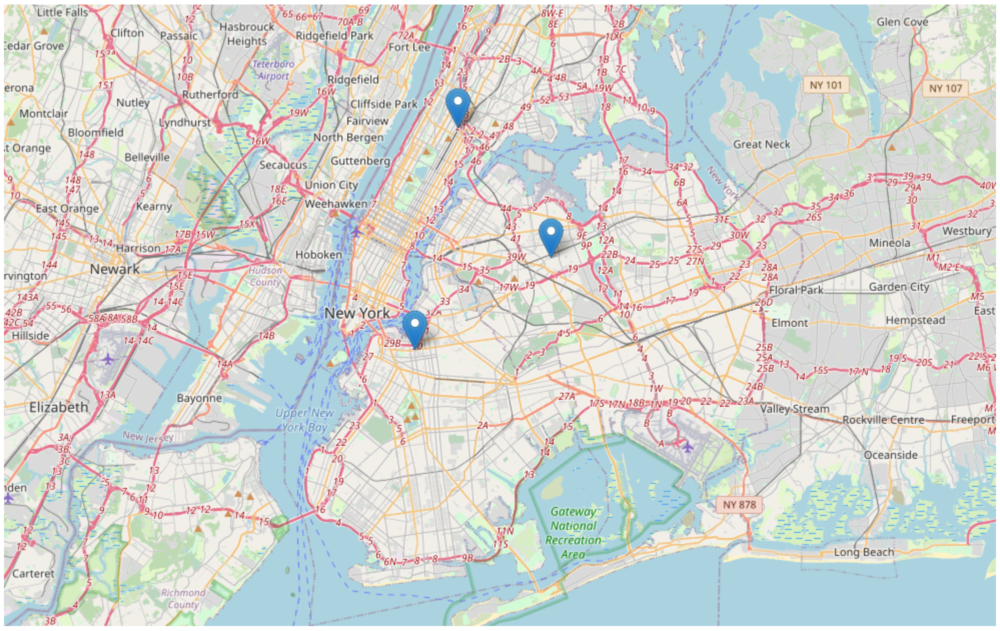
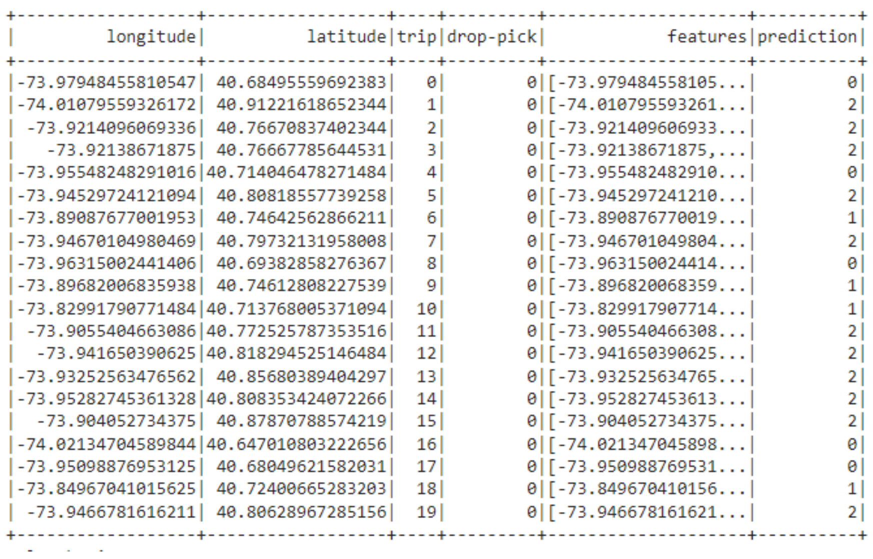


Figure 13: Cluster centers for concatenated coordinates.



```
+------------------+------------------+----+---------+--------------------+----------+
|         longitude|          latitude|trip|drop-pick|            features|prediction|
+------------------+------------------+----+---------+--------------------+----------+
|-73.97948455810547| 40.68495559692383|   0|        0|[-73.979484558105...|         0|
|-74.01079559326172| 40.91221618652344|   1|        0|[-74.010795593261...|         2|
| -73.9214096069336| 40.76670837402344|   2|        0|[-73.921409606933...|         2|
|   -73.92138671875| 40.76667785644531|   3|        0|[-73.92138671875,...|         2|
|-73.95548248291016|40.714046478271484|   4|        0|[-73.955482482910...|         0|
|-73.94529724121094| 40.80818557739258|   5|        0|[-73.945297241210...|         2|
|-73.89087677001953| 40.74642562866211|   6|        0|[-73.890876770019...|         1|
|-73.94670104980469| 40.79732131958008|   7|        0|[-73.946701049804...|         2|
|-73.96315002441406| 40.69382858276367|   8|        0|[-73.963150024414...|         0|
|-73.89682006835938| 40.74612808227539|   9|        0|[-73.896820068359...|         1|
|-73.82991790771484|40.713768005371094|  10|        0|[-73.829917907714...|         1|
| -73.9055404663086|40.772525787353516|  11|        0|[-73.905540466308...|         2|
|  -73.941650390625|40.818294525146484|  12|        0|[-73.941650390625...|         2|
|-73.93252563476562| 40.85680389404297|  13|        0|[-73.932525634765...|         2|
|-73.95282745361328|40.808353424072266|  14|        0|[-73.952827453613...|         2|
|  -73.904052734375| 40.87870788574219|  15|        0|[-73.904052734375...|         2|
|-74.02134704589844|40.647010803222656|  16|        0|[-74.021347045898...|         0|
|-73.95098876953125| 40.68049621582031|  17|        0|[-73.950988769531...|         0|
|-73.84967041015625| 40.72400665283203|  18|        0|[-73.849670410156...|         1|
| -73.9466781616211| 40.80628967285156|  19|        0|[-73.946678161621...|         2|
+------------------+------------------+----+---------+--------------------+----------+
```

Figure 14: Dataframe for Predicted Trips.

Below we show the count and percentages for each of the three clusters [0,1,2]. This table shows the count and percentage of trips both leaving (pickup in cluster) and coming (dropoff in cluster).

```
+----------+-------+------------------+
|prediction|  count|        percentage|
+----------+-------+------------------+
|         0|1169668|39.214363638436154|
|         1| 584750| 19.60436562988433|
|         2|1228336| 41.18127073167951|
+----------+-------+------------------+
```

This table shows the count and percentage of trips whose pickup is in this cluster.

```
+----------+------+------------------+
|prediction| count|        percentage|
+----------+------+------------------+
|         0|581872|39.015755238279795|
|         1|277694|18.619973353484735|
|         2|631811|42.364271408235474|
+----------+------+------------------+
```

This table shows the count and percentage of trips whose dropoff is in this cluster.

```
+----------+------+------------------+
|prediction| count|        percentage|
+----------+------+------------------+
|         0|587796| 39.41297203859252|
|         1|307056|20.588757906283924|
|         2|596525|39.998270055123555|
+----------+------+------------------+
```

We then at the end calculate that the pickup and dropoff in same cluster percentage is 85.547%

2.7 Task 7

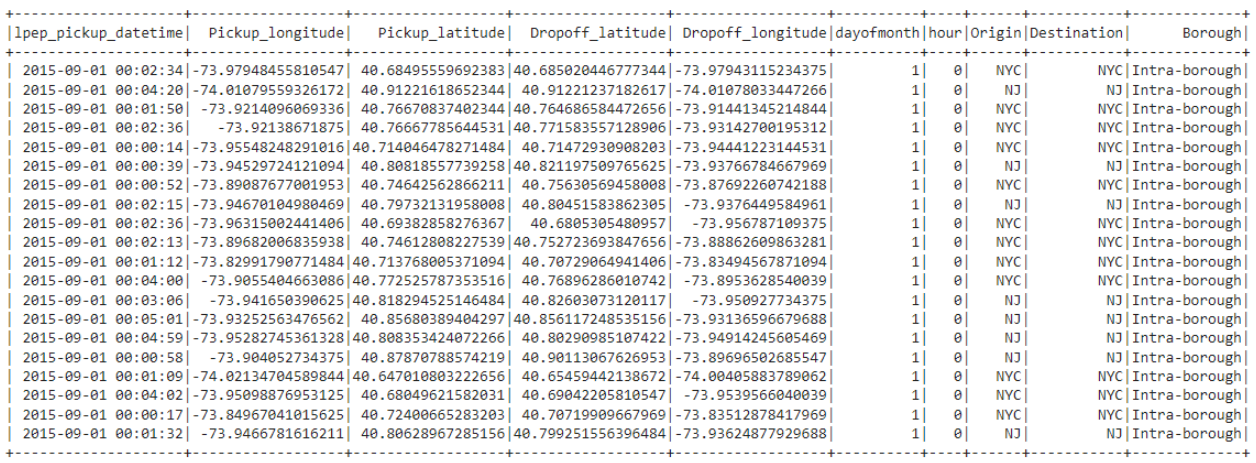
Since the Hudson River, which is the boundary of New Jersey and NYC in this area, is quite straight, we can use a line to model this natural boundary. The approximated line in latitude (y) and longitude (x) can be represented as:

```
y = 1.323942 * x + 138.669195
```

If a location satisfies y > 1.323942x + 138.669195, it's in New Jersey. If a location satisfies y < 1.323942x + 138.669195, it's in Manhattan. After such processing, we can get Figure 2. In this way, we can better utilize the pick-up and drop-off latitude-longitude data.

In this task, you are expected to group all trips into the following four categories, NJ → NJ, NJ → NYC, NYC → NJ, NYC → NYC. Can you build some association rules on intra- vs. inter-borough traffic? What story does it tell about how New Yorkers use their green taxis? For example, in which hour of the day, there would be more inter-borough traffic than intra-borough traffic? For those NYC → NYC trips, people are more likely to take taxis from uptown to downtown or from downtown to uptown?

In this task we aim to build some association rules and do some analysis based on the equation which tells us that when y > 1.323942x + 138.669195, it's New Jersey and when y < 1.323942x + 138.669195, it's Manhattan. Based on that information we derive extra columns such as the origin and destination based on the what the pickup longitude and latitude are and what the dropoff longitude and latitude are. We then using the origin and destination check whether the origin to destination is an inter-borough trip or intra-borough trip. These extra features are shown in Figure 15.



```
+--------------------+------------------+------------------+------------------+------------------+----------+----+------+-----------+-------------+
|lpep_pickup_datetime|  Pickup_longitude|   Pickup_latitude|  Dropoff_latitude| Dropoff_longitude|dayofmonth|hour|Origin|Destination|      Borough|
+--------------------+------------------+------------------+------------------+------------------+----------+----+------+-----------+-------------+
| 2015-09-01 00:02:34|-73.97948455810547| 40.68495559692383|40.685020446777344|-73.97943115234375|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:04:20|-74.01079559326172| 40.91221618652344| 40.91221237182617|-74.01078033447266|         1|   0|    NJ|         NJ|Intra-borough|
| 2015-09-01 00:01:50| -73.9214096069336| 40.76670837402344|40.764686584472656|-73.91441345214844|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:02:36|   -73.92138671875| 40.76667785644531|40.771583557128906|-73.93142700195312|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:00:14|-73.95548248291016|40.714046478271484| 40.71472930908203|-73.94441223144531|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:00:39|-73.94529724121094| 40.80818557739258|40.821197509765625|-73.93766784667969|         1|   0|    NJ|         NJ|Intra-borough|
| 2015-09-01 00:00:52|-73.89087677001953| 40.74642562866211| 40.75630569458008|-73.87692260742188|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:02:15|-73.94670104980469| 40.79732131958008| 40.80451583862305| -73.9376449584961|         1|   0|    NJ|         NJ|Intra-borough|
| 2015-09-01 00:02:36|-73.96315002441406| 40.69382858276367|  40.6805305480957|  -73.956787109375|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:02:13|-73.89682006835938| 40.74612808227539|40.752723693847656|-73.88862609863281|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:01:12|-73.82991790771484|40.713768005371094| 40.70729064941406|-73.83494567871094|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:04:00| -73.9055404663086|40.772525787353516| 40.76896286010742| -73.8953628540039|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:03:06|  -73.941650390625|40.818294525146484| 40.82603073120117|  -73.950927734375|         1|   0|    NJ|         NJ|Intra-borough|
| 2015-09-01 00:05:01|-73.93252563476562| 40.85680389404297|40.856117248535156|-73.93136596679688|         1|   0|    NJ|         NJ|Intra-borough|
| 2015-09-01 00:04:59|-73.95282745361328|40.808353424072266| 40.80290985107422|-73.94914245605469|         1|   0|    NJ|         NJ|Intra-borough|
| 2015-09-01 00:00:58|  -73.904052734375| 40.87870788574219| 40.90113067626953|-73.89696502685547|         1|   0|    NJ|         NJ|Intra-borough|
| 2015-09-01 00:01:09|-74.02134704589844|40.647010803222656| 40.65459442138672|-74.00405883789062|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:04:02|-73.95098876953125| 40.68049621582031| 40.69042205810547| -73.9539566040039|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:00:17|-73.84967041015625| 40.72400665283203| 40.70719909667969|-73.83512878417969|         1|   0|   NYC|        NYC|Intra-borough|
| 2015-09-01 00:01:32| -73.9466781616211| 40.80628967285156|40.799251556396484|-73.93624877929688|         1|   0|    NJ|         NJ|Intra-borough|
+--------------------+------------------+------------------+------------------+------------------+----------+----+------+-----------+-------------+
```

Figure 15: Dataframe for Association Mining

We then also use a concept from Association mining where looking at the support of the origin and destination, we attempt to calculate the confidence of travelling from Borough A to Borough B.

For NYC -> NYC 85.733%
Support(NYC,NYC)/Support(NYC)

```
(temp.filter(col('Origin') == 'NYC').filter(col('Destination') == 'NYC').count()
/temp.filter(col('Origin') == 'NYC').count())*100
```

For NYC -> NJ 14.266%
Support(NYC,NJ)/Support(NYC)

```
(temp.filter(col('Origin') == 'NYC').filter(col('Destination') == 'NJ').count()
/temp.filter(col('Origin') == 'NYC').count())*100
```

For NJ -> NJ 92.455%
Support(NJ,NJ)/Support(NJ)

```
(temp.filter(col('Origin') == 'NJ').filter(col('Destination') == 'NJ').count()
/temp.filter(col('Origin') == 'NJ').count())*100
```

For NJ -> NYC 7.545%
Support(NJ,NYC)/Support(NJ)

```
(temp.filter(col('Origin') == 'NJ').filter(col('Destination') == 'NYC').count()
/temp.filter(col('Origin') == 'NJ').count())*100
```

We then look at the hour of the day, and see when would be more inter-borough traffic than intra-borough traffic by taking a groupby of the hour and the borough and seeing the count of each which clearly shows at which hour are more people commuting inter or intra- borough.

```
+----+-------------+-----+
|hour|      Borough|count|
+----+-------------+-----+
|   0|Inter-borough| 8217|
|   0|Intra-borough|58789|
 ------------------------
|   1|Intra-borough|47188|
|   1|Inter-borough| 6474|
 ------------------------
|   2|Inter-borough| 4870|
|   2|Intra-borough|36235|
 ------------------------
|   3|Intra-borough|27642|
|   3|Inter-borough| 3922|
 ------------------------
|   4|Intra-borough|22821|
|   4|Inter-borough| 3529|
 ------------------------
|   5|Inter-borough| 3151|
|   5|Intra-borough|13488|
 ------------------------
|   6|Inter-borough| 4755|
|   6|Intra-borough|17846|
 ------------------------
|   7|Inter-borough| 7419|
|   7|Intra-borough|34454|
 ------------------------
|   8|Intra-borough|48678|
|   8|Inter-borough|10123|
 ------------------------
|   9|Intra-borough|51709|
|   9|Inter-borough|10156|
 ------------------------
|  10|Inter-borough| 8031|
|  10|Intra-borough|49272|
 ------------------------
|  11|Intra-borough|49460|
|  11|Inter-borough| 7157|
 ------------------------
|  12|Intra-borough|50818|
|  12|Inter-borough| 6831|
 ------------------------
```

```
|  13|Intra-borough|51041|
|  13|Inter-borough| 6265|
 ------------------------
|  14|Inter-borough| 6922|
|  14|Intra-borough|59551|
 ------------------------
|  15|Intra-borough|66210|
|  15|Inter-borough| 7377|
 ------------------------
|  16|Inter-borough| 7392|
|  16|Intra-borough|71561|
 ------------------------
|  17|Inter-borough| 8086|
|  17|Intra-borough|79749|
 ------------------------
|  18|Intra-borough|88150|
|  18|Inter-borough| 8909|
 ------------------------
|  19|Intra-borough|86674|
|  19|Inter-borough| 9299|
 ------------------------
|  20|Intra-borough|81308|
|  20|Inter-borough| 9307|
 ------------------------
|  21|Inter-borough|10339|
|  21|Intra-borough|76036|
 ------------------------
|  22|Inter-borough|11136|
|  22|Intra-borough|73400|
 ------------------------
|  23|Intra-borough|69180|
|  23|Inter-borough|10450|
+----+-------------+-----+
```

2.8 Task 8

Could you suggest to Green Taxi how to improve their business? Think about ways they could decrease cost, get more customers, improve quality of service etc. In all tasks, try to use the algorithms that you learnt in class, if you didn't succeed, you can come up with other tasks yourself using this data set.

As we can see from the Inter-borough and Intra-borough distribution above for hourly trips that their are more trips which are Intra-borough that means (NYC -> NYC) and (NJ -> NJ). This can further be proved by the association mining rules and confidences we came up with. In it we could see that the given the person's origin was NYC there was a 85.73% confidence that the traveller will go within NYC. Same case for NJ that given the traveller's origin is NJ there is a 92.45% confidence that the traveller will travel within NJ. We also from our clustering can see that there are 3 main clusters where the people are picked up and dropped off and that those 3 cluster centers are almost the same for the pickup and dropoffs. Finally, we can see that the hourly trips made by each taxi within the hour is around 3 to 4km and from the histogram we can see that the distance is rarely more than 10kms.

Looking at all this data a way to decrease cost, get more customers, improve quality of service is to keep green taxis and keep premium green taxis. The premium green taxis would be those taxis that would be go from one borough to the other and they could have a fixed pick up and dropoff place. The people who would like to travel across boroughs could take a green taxi from their current position and to the fixed pickup and dropoff stations and from there they can travel across boroughs. This may cause hindrance to the users but an incentive would be to decrease the cost of the ride thus more customers would prefer to take the premium taxi across borughs. This in turn would also encourage more customers to travel because of the cheap prices. Finally, it would also decrease the cost of the taxi's when multiple customers will use the same taxi to travel across a borough thus saving multiple trips and decreasing the cost.

References

[1] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

[2] https://www1.nyc.gov/assets/tlc/downloads/pdf/datadictionarytriprecordsgreen.pdf

[3] https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb