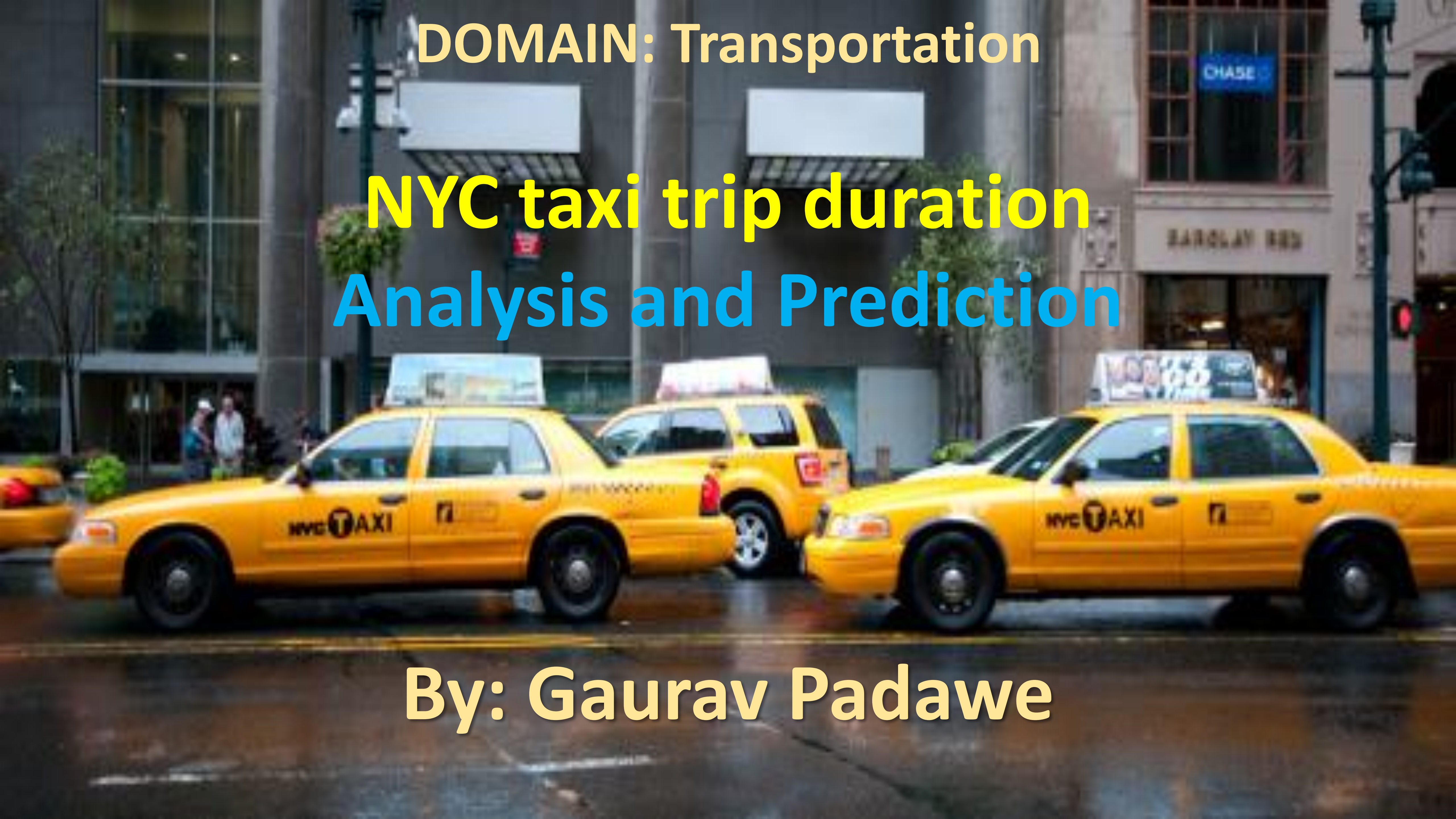
DOMAIN: Transportation

NYC taxi trip duration

Analysis and Prediction



By: Gaurav Padawe

Presentation Outline

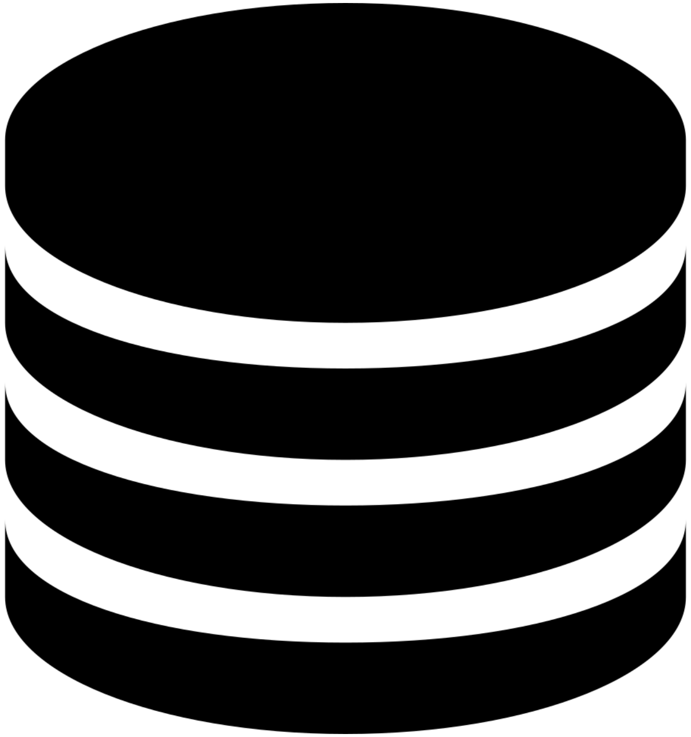
- Business Objective
- Data Source
- Methodology
- Evaluation
- Summary

Business Objective

Build a model that predicts the total trip duration of taxi trips in New York City.

Data Source:

Source: NYC Taxi and Limousine Commission (TLC) and Kaggle



Statistics:

- **Rows - 1458644**
- **Features - 11 (Including Target)**
- **Target – Trip Duration**

Questions Arising

- What story Spatial data has for us ?
- Are there any Missing Values ?
- What types of Variables do we have ?
- Do we’ve False trips which exceeds Trip Duration well above impossibility ?
- What’s the most frequent travelled destination ?

And much more to know as we continue with Analysis.

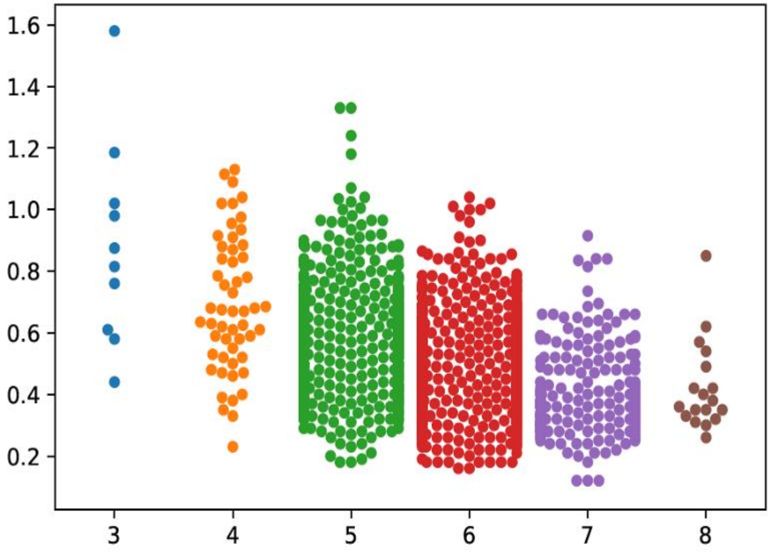
Methodology

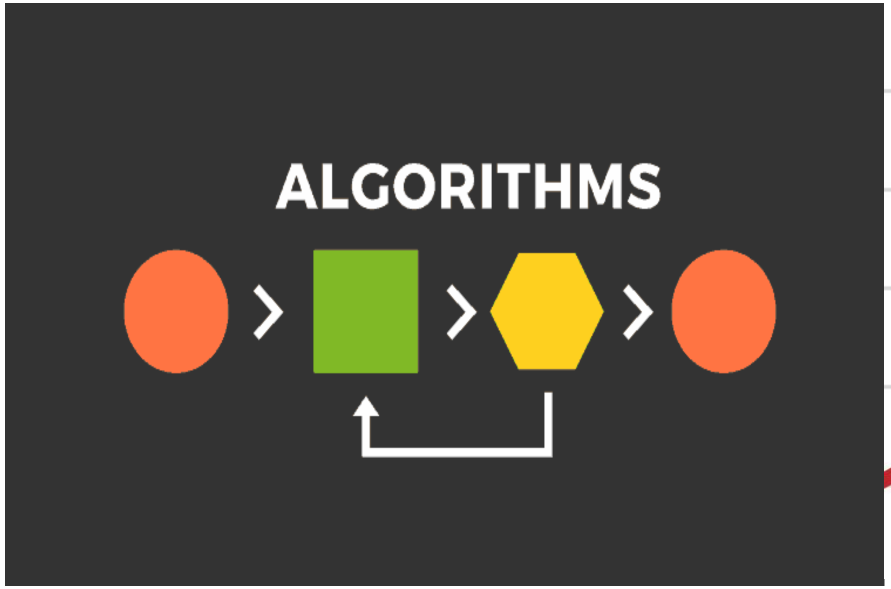
Approach

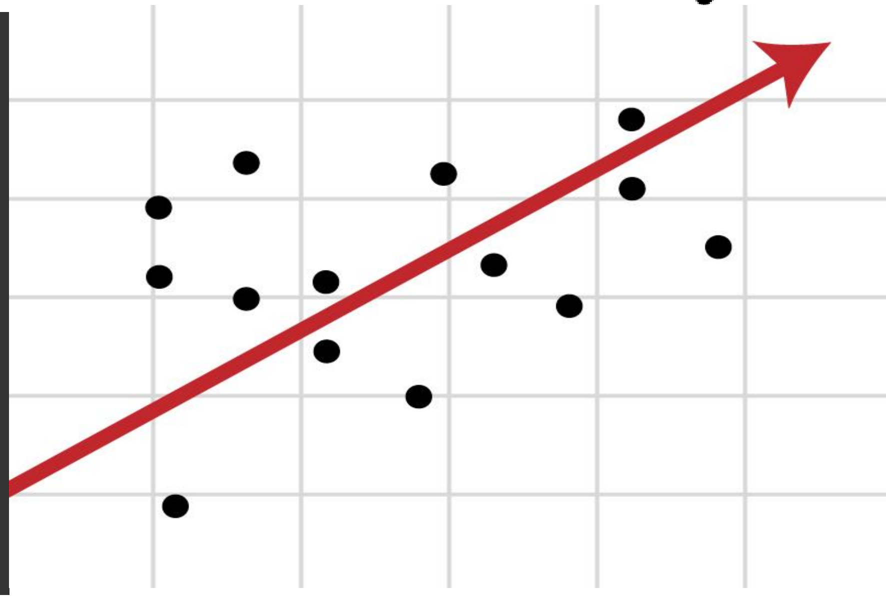
Data Preparation
and
Exploratory Data Analysis

Building Predictive Model
using Multiple
Techniques / Algorithms

Optimal Model Identified
through
Testing and Evaluation







Machine Learning Algorithm

- PCA
- Linear Regression
- Decision Tree
- Random Forest

Tools Used

- Jupyter Notebook(Python)
- Tableau
- Google Collab Research

Data Preparation and EDA

- First, begin with setting our path and importing required packages.

```
In [0]: path = "D:/Data Science/DS Prac/ML Algo/ML Project datasets/Project datasets modified/NYC Taxi Trip/NYC Taxi Trip/"
```

```
In [0]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
```

- Now, Let's read our dataset, we have some good amount of features in it.

```
In [4]: nyc_taxi = pd.read_csv(path+"train.csv", header=0, parse_dates=True)
        nyc_taxi.head()
```

Out[4]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982155	40.767937	-73.964630	40.765602	
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980415	40.738564	-73.999481	40.731152	
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979027	40.763939	-74.005333	40.710087	
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.010040	40.719971	-74.012268	40.706718	
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.973053	40.793209	-73.972923	40.782520	

- We have some features with “object” dtype and quite surprising to see that no Null Values.

- **Data fields:**
  - id - a unique identifier for each trip
  - vendor_id - a code indicating the provider associated with the trip record
  - pickup_datetime - date and time when the meter was engaged
  - dropoff_datetime - date and time when the meter was disengaged
  - passenger_count - the number of passengers in the vehicle (driver entered value)
  - pickup_longitude - the longitude where the meter was engaged
  - pickup_latitude - the latitude where the meter was engaged
  - dropoff_longitude - the longitude where the meter was disengaged
  - dropoff_latitude - the latitude where the meter was disengaged
  - store_and_fwd_flag - This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip
  - trip_duration - duration of the trip in seconds

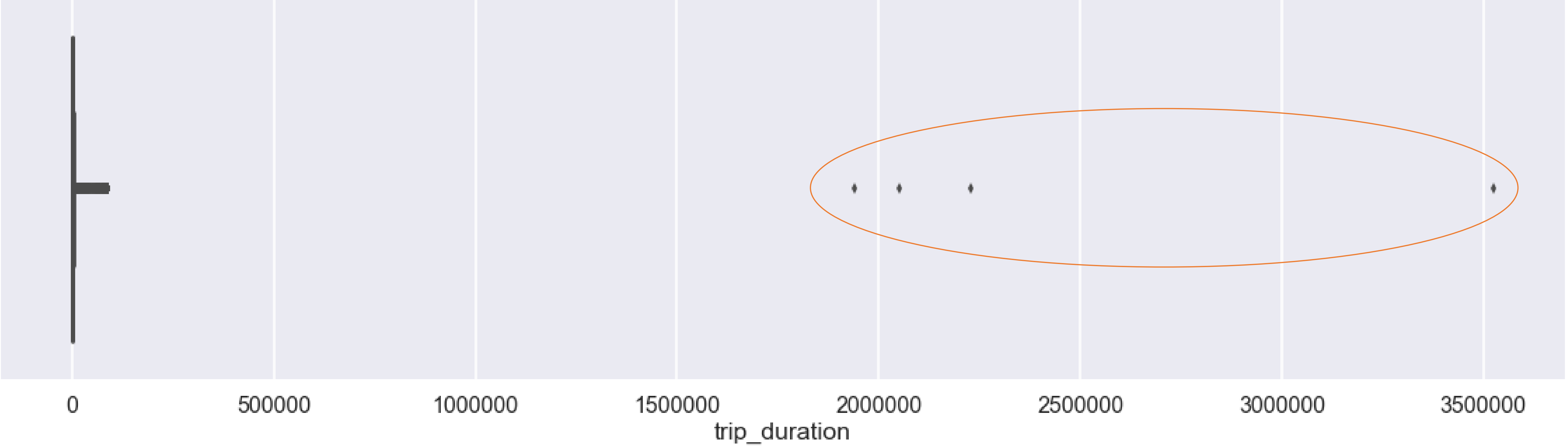
```
In [5]: nyc_taxi.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1458644 entries, 0 to 1458643
Data columns (total 11 columns):
id                    1458644 non-null object
vendor_id             1458644 non-null int64
pickup_datetime       1458644 non-null object
dropoff_datetime      1458644 non-null object
passenger_count       1458644 non-null int64
pickup_longitude      1458644 non-null float64
pickup_latitude       1458644 non-null float64
dropoff_longitude     1458644 non-null float64
dropoff_latitude      1458644 non-null float64
store_and_fwd_flag    1458644 non-null object
trip_duration         1458644 non-null int64
dtypes: float64(4), int64(3), object(4)
memory usage: 122.4+ MB
```

- Now, we will perform basic EDA to get some insights into the data.

Vendor_ID stating the provider associated with trip, preferably 2 different taxi companies.
Analysis tells us that Second service provider has been most frequently opted by people over First service provider over the period of time.

We have almost all the trip records sent to the server.

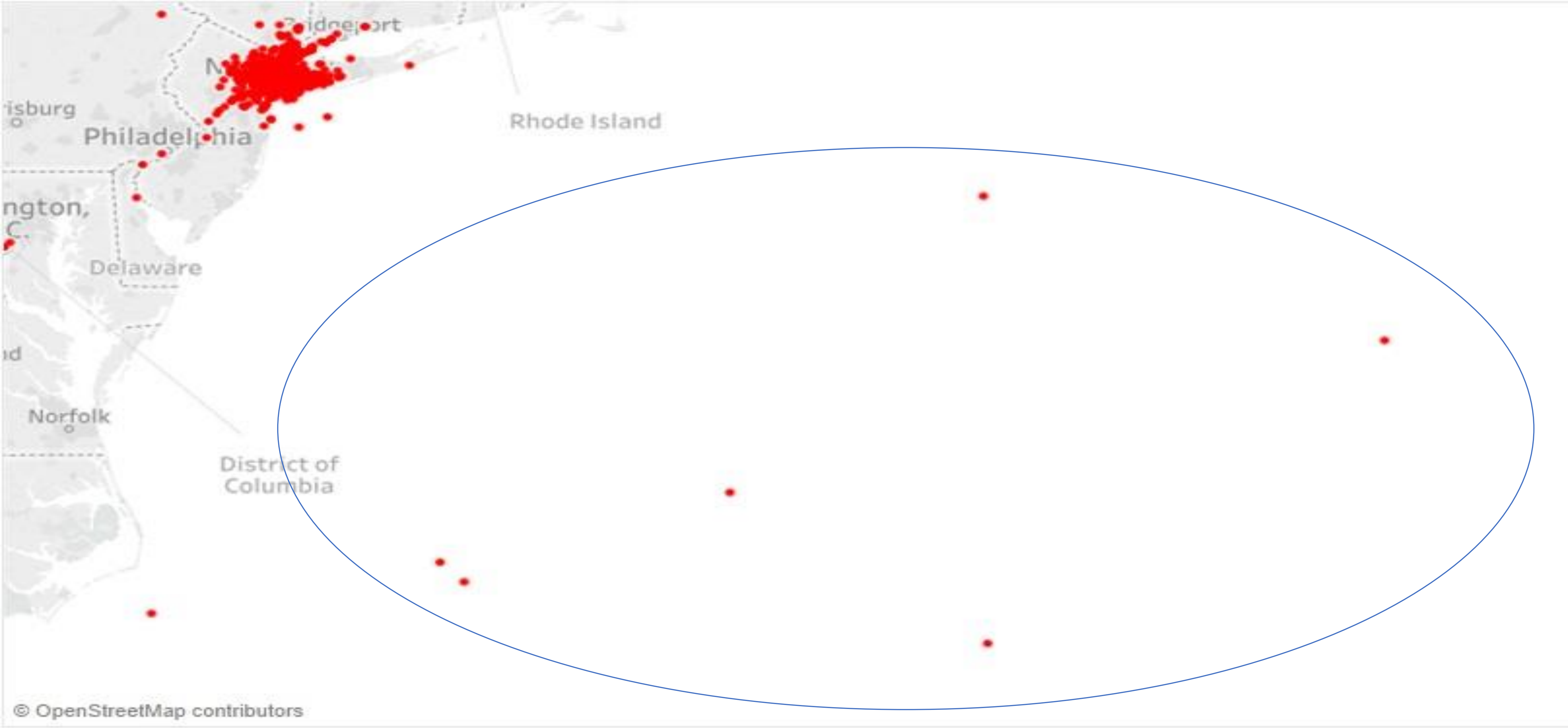


Trip Duration Viz.

Probably in this visualization we can clearly see some outliers (marked in Red) , their trips are lasting between 1900000 seconds (528 Hours) to somewhere around 3500000 (972 hours) seconds which is impossible in case of taxi trips , How can a taxi trip be that long ? It's Quite suspicious. We'll have to get rid of those Outliers or else it'll affect our model's performance.

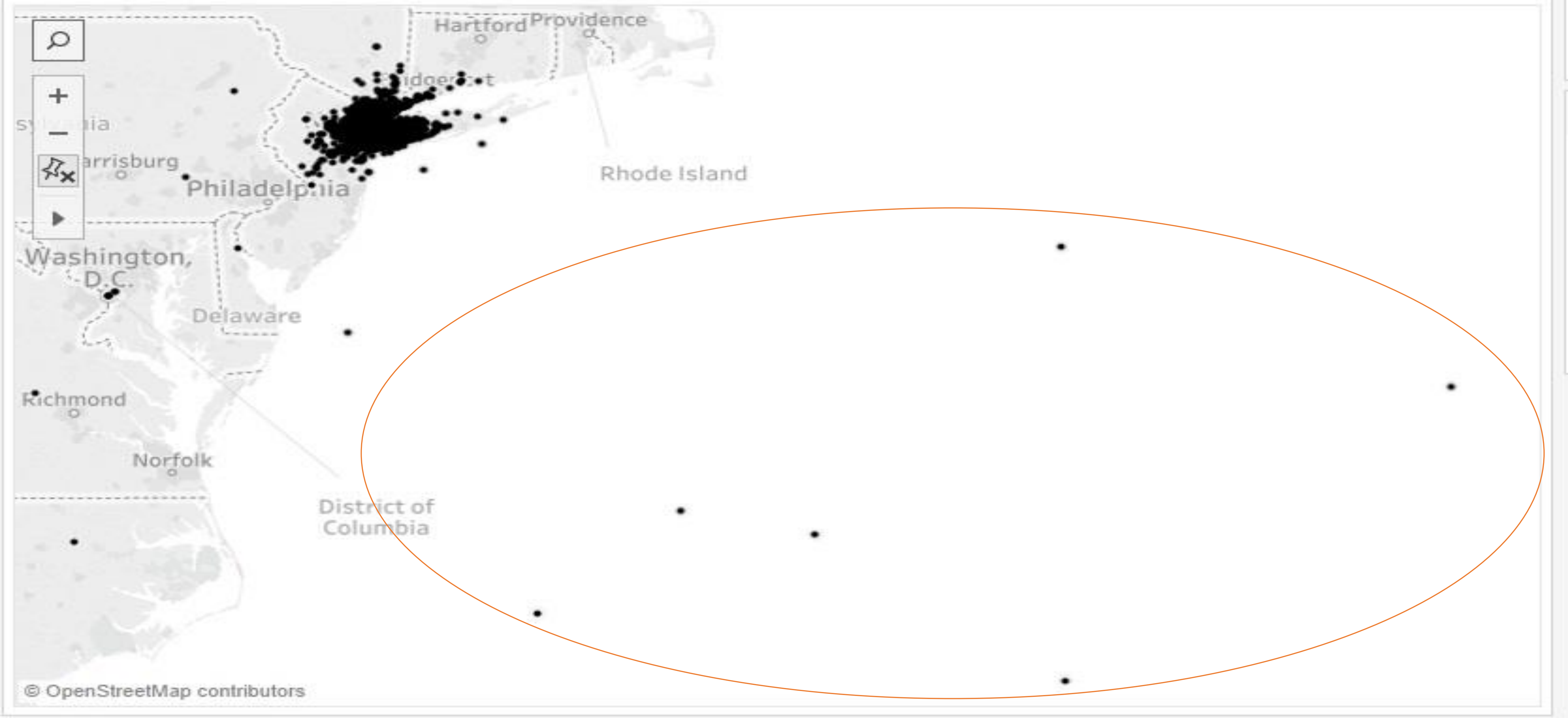
Pickup Points over the period of time, Apart from Manhattan we've some areas where we see most pickups, The LA Guardia Airport and JFK Airport.

[Click here] for Interactive Visualization



Some the pickup points reach far beyond the Land , probably in Sea which is kind of impossible, how can a taxi trip begin in Ocean ? Curiousity rises !

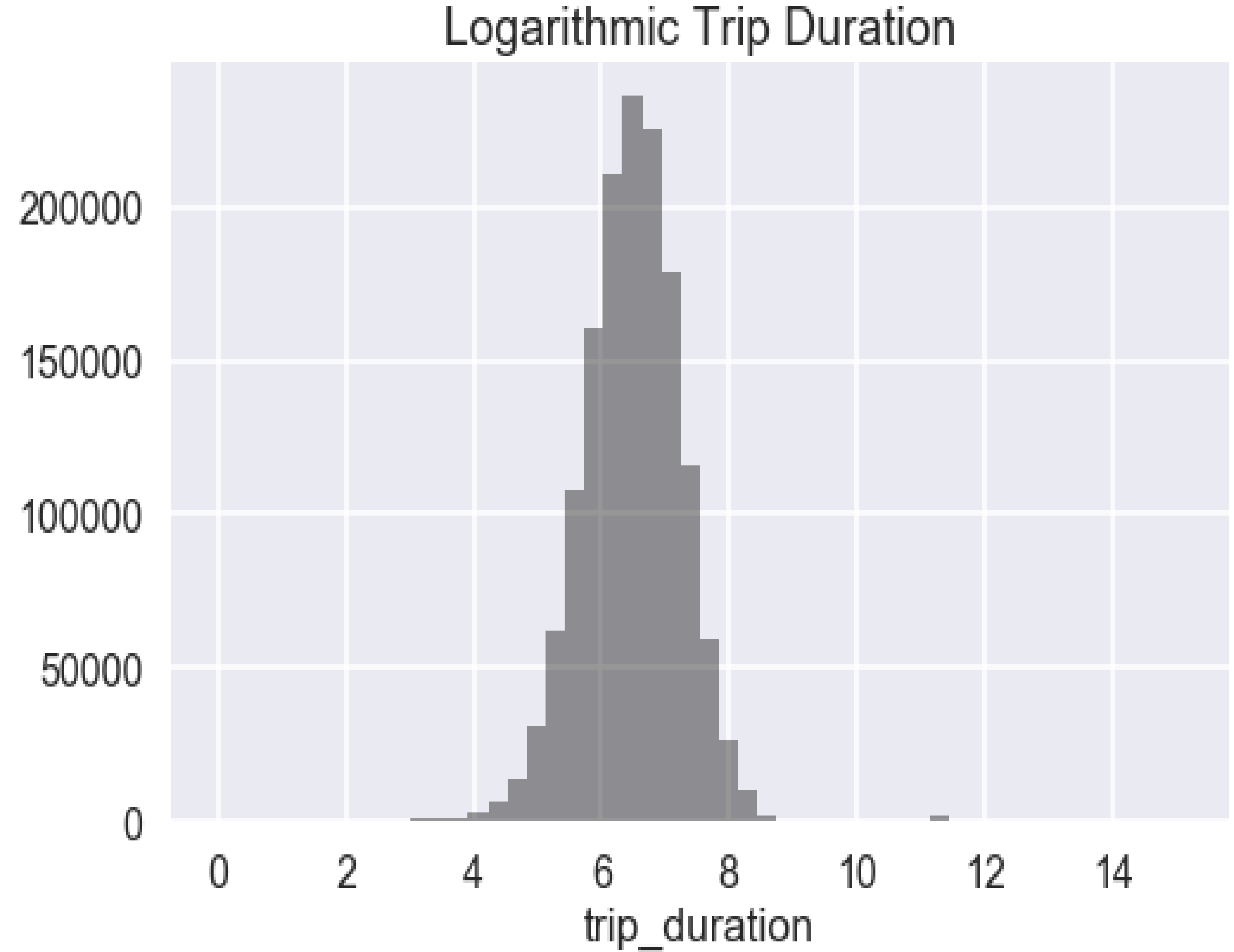
DropOff Points Visualization: Leaving Manhattan area, the place where we see more DropOffs are Airports (Marked in Red) , **The LA Guardia Airport and JFK Airport.**



It's evident from the above marked pickup and dropoff points which lead in the North Atlantic Sea, maybe these points responsible for 350000 seconds (972 Hours) trip duration, possibly an outliers.

Most number of trips are done by single or double passengers.

But one thing is Interesting to observe, there exist trip with ZERO passengers, was that a free ride ? Or just a False data recorded ?



By Taking Logarithm of Trip duration we can even out those outliers which could had possibly affected our model. Now, Log. Trip Duration is our Target to Predict.

Label Encoding Categorical Variables ,i.e, “store_and_fwd_flag” and “vendor_id”. We can convert these features into “category” type by function called “astype(‘category’)” that will speed up the Computation. Since, my plan is to go with PCA for dimension reduction, I’m not going with that approach.

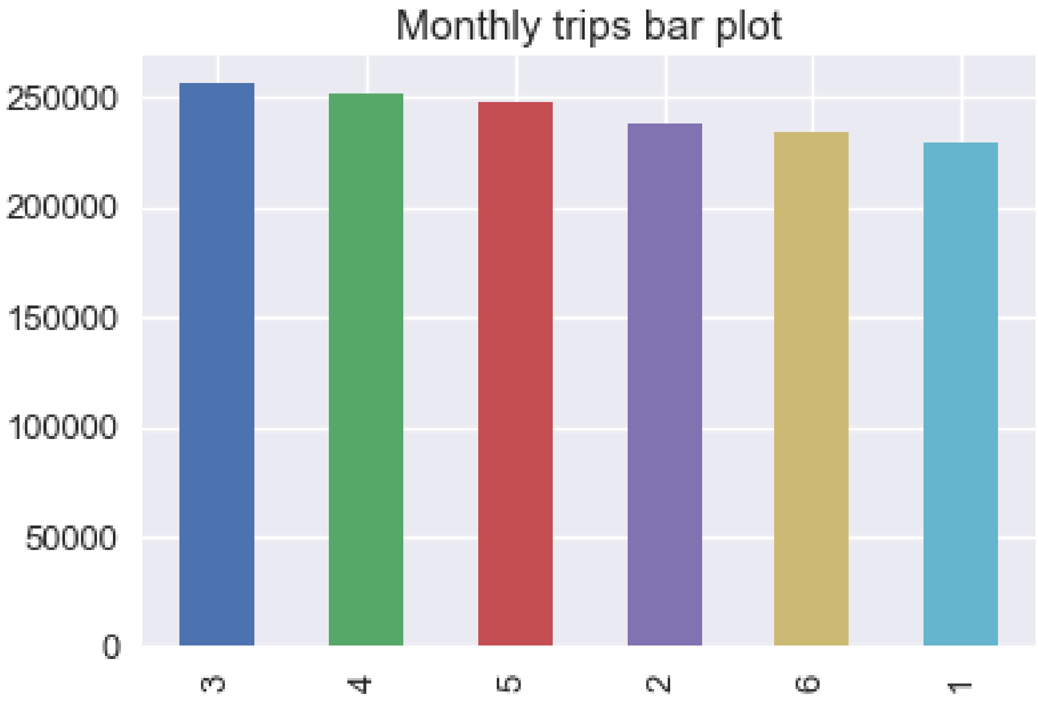
```
from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()
nyc_taxi['store_and_fwd_flag'] = enc.fit_transform(nyc_taxi['store_and_fwd_flag'])
nyc_taxi['vendor_id'] = enc.fit_transform(nyc_taxi['vendor_id'])
```

Feature Engineering

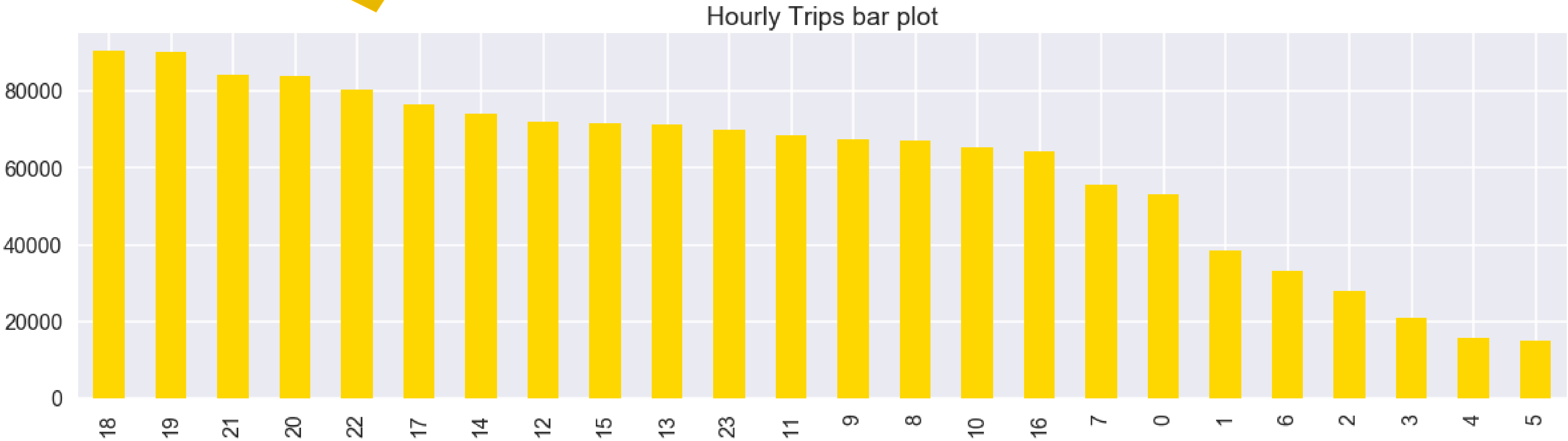
The Date and time columns in the Dataset has whole lot story to tell, we have to fetch them as separate columns. We do not have to fetch pickup and dropoff time both, as they may lead to strong positive co-relation in the respective fetched features. Further we can use these columns for Analysis.

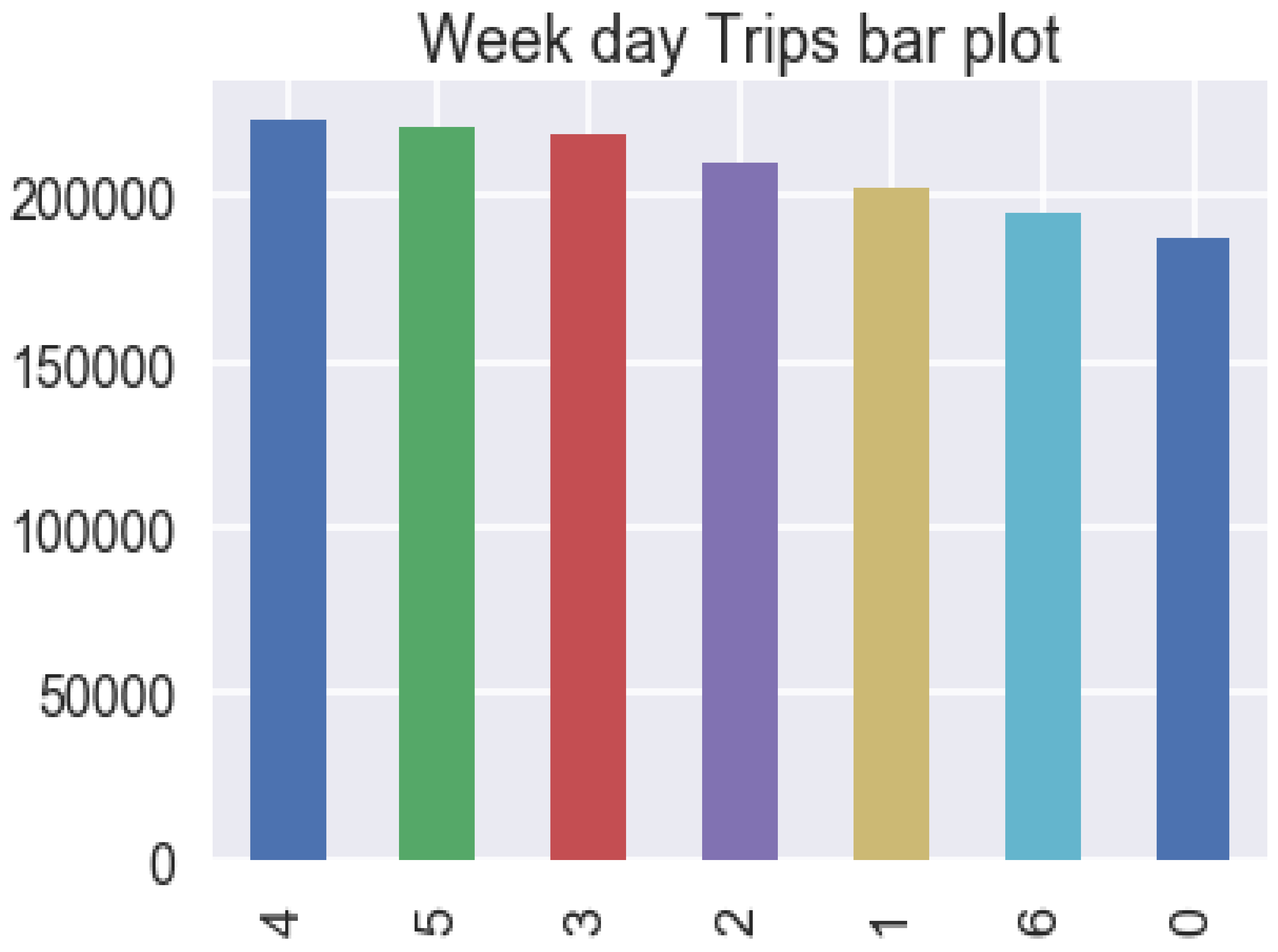
```
nyc_taxi['pickup_datetime'] = pd.to_datetime(nyc_taxi['pickup_datetime'])
nyc_taxi['dropoff_datetime'] = pd.to_datetime(nyc_taxi['dropoff_datetime'])

nyc_taxi['pickup_day'] = nyc_taxi['pickup_datetime'].dt.day
nyc_taxi['pickup_month'] = nyc_taxi['pickup_datetime'].dt.month
nyc_taxi['pickup_date'] = nyc_taxi['pickup_datetime'].dt.date
nyc_taxi['pickup_hour'] = nyc_taxi['pickup_datetime'].dt.hour
nyc_taxi['pickup_min'] = nyc_taxi['pickup_datetime'].dt.minute
nyc_taxi['pickup_weekday'] = nyc_taxi['pickup_datetime'].dt.weekday

nyc_taxi['dropoff_min'] = nyc_taxi['dropoff_datetime'].dt.minute
```

Number of trips in a particular month - March and April marking the highest. January being lowest maybe due to SnowFall.

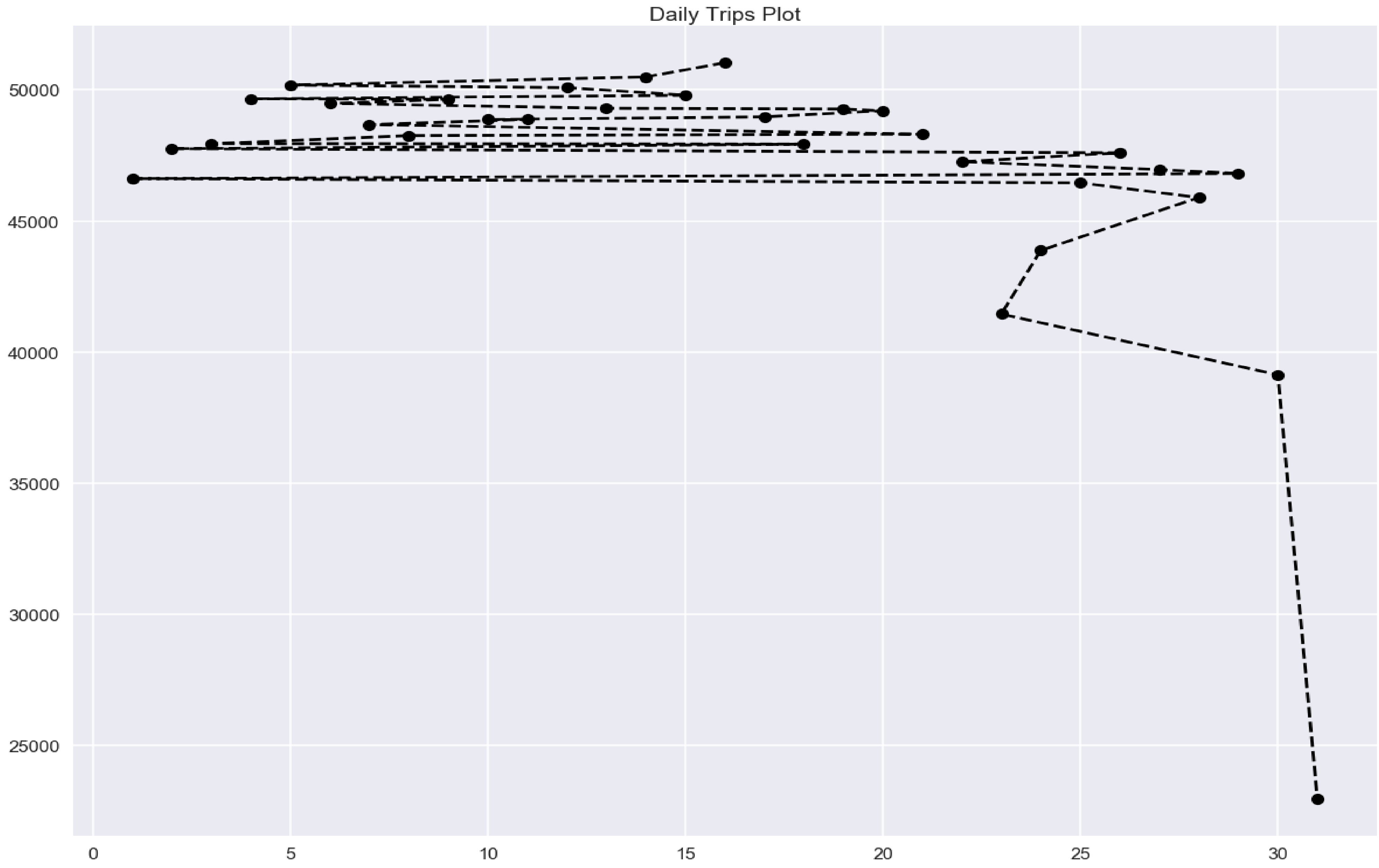


In which hour we get to see maximum trips ? - Rush hours (4 pm to 8 pm)



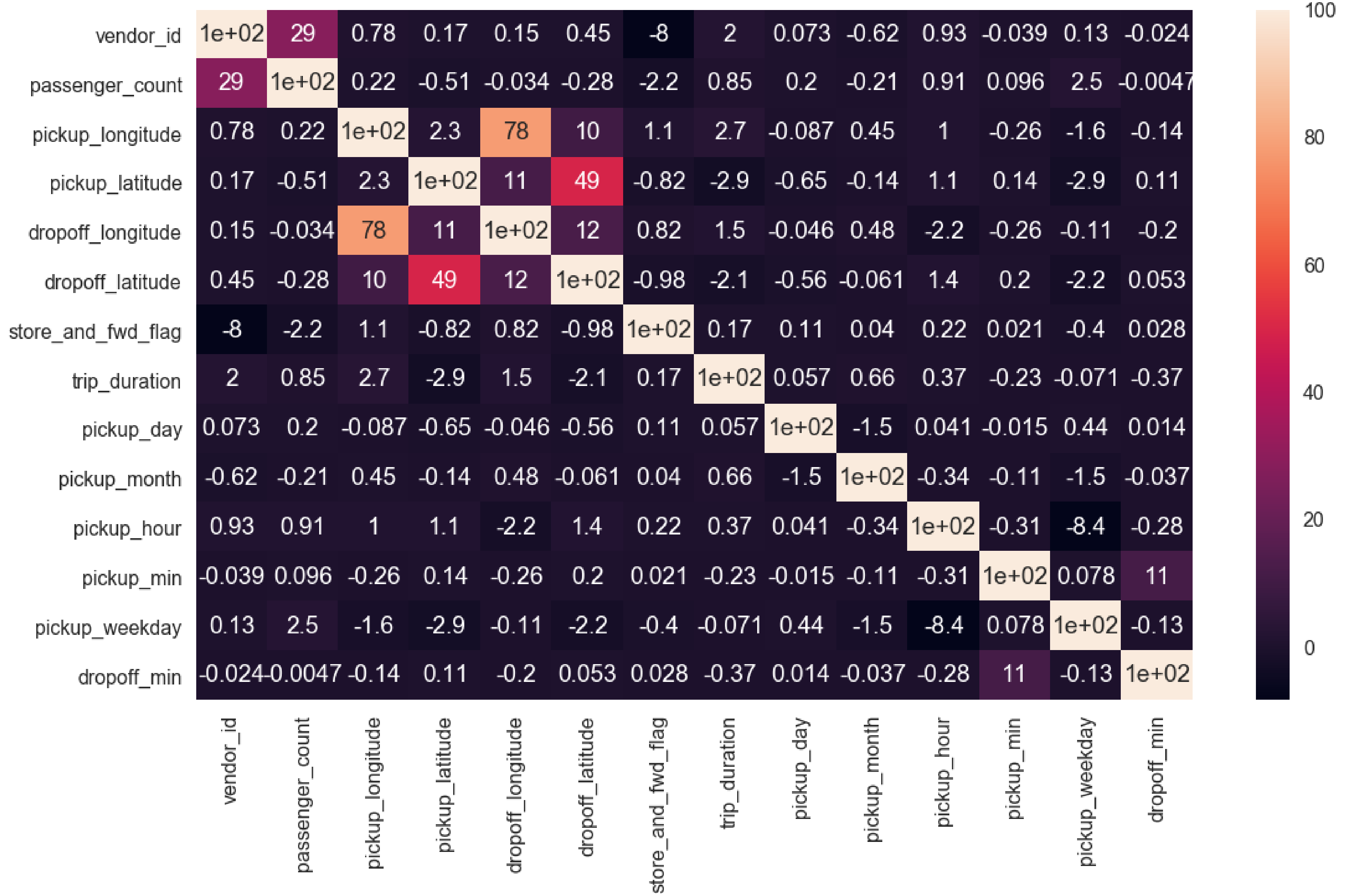


Observations says that Friday's and Saturday's are those days in a week when New Yorkers prefer to get out of their home. GREAT !!



Seem like New Yorker’s do not prefer to get a Taxi on Month end’s , there is a significant drop in the Taxi trip count as month end’s approach.

Co-relation Heatmap



Let's drop unwanted columns like ID, as it makes no sense and some other columns of which we have already fetched information separately.

In [22]:

```
#Droping Unwanted Columns

nyc_taxi = nyc_taxi.drop(['id','pickup_datetime','pickup_date','dropoff_datetime'], axis=1)
nyc_taxi.head()
```

Out[22]:

	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	trip_duration	pickup_day	pickup_mc
0	1	1	-73.982155	40.767937	-73.964630	40.765602	0	455	14	
1	0	1	-73.980415	40.738564	-73.999481	40.731152	0	663	12	
2	1	1	-73.979027	40.763939	-74.005333	40.710087	0	2124	19	
3	1	1	-74.010040	40.719971	-74.012268	40.706718	0	429	6	
4	1	1	-73.973053	40.793209	-73.972923	40.782520	0	435	26	

Normalizing the Dataset using Standard Scaling Technique.

Now, Why Standard Scaling ? Why not MinMax or Normalizer ?

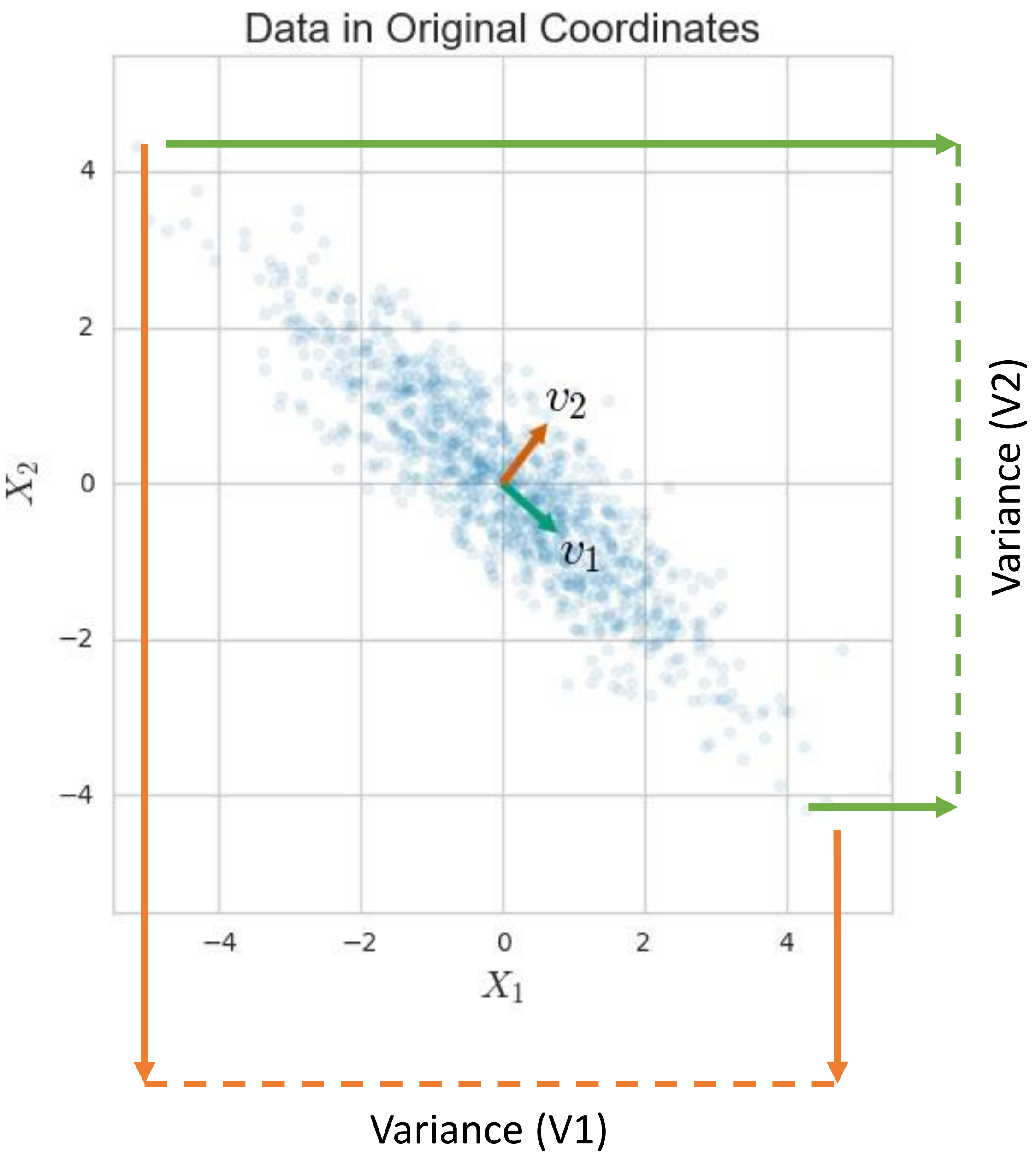
It is because MinMax adjusts the value between 0’s and 1’s ,
which tend to work better for optimization techniques like Gradient descent and machine learning algorithms like KNN.

While, Normalizer uses distance measurement like Euclidean or Manhattan, so Normalizer tend to work better with KNN.

```
from sklearn.preprocessing import StandardScaler, Normalizer, MinMaxScaler

cols = X.columns

ss = StandardScaler()
#norm = Normalizer()
#mms = MinMaxScaler()

new_df = ss.fit_transform(X)
new_df = pd.DataFrame(new_df, columns=cols)
new_df.head()
```

	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag	pickup_day	pickup_month	pickup_h
0	0.932380	-0.505637	-0.122261	0.517494	0.124369	0.384575	-0.074471	-0.172813	-0.307440	0.530
1	-1.072524	-0.505637	-0.097727	-0.375819	-0.368970	-0.575303	-0.074471	-0.402616	1.477173	-2.126
2	0.932380	-0.505637	-0.078143	0.395910	-0.451805	-1.162220	-0.074471	0.401692	-1.497182	-0.407
3	0.932380	-0.505637	-0.515558	-0.941274	-0.549976	-1.256071	-0.074471	-1.092023	0.287431	0.842
4	0.932380	-0.505637	0.006112	1.286091	0.006974	0.855957	-0.074471	1.206001	-0.307440	-0.094

Principal Component Analysis

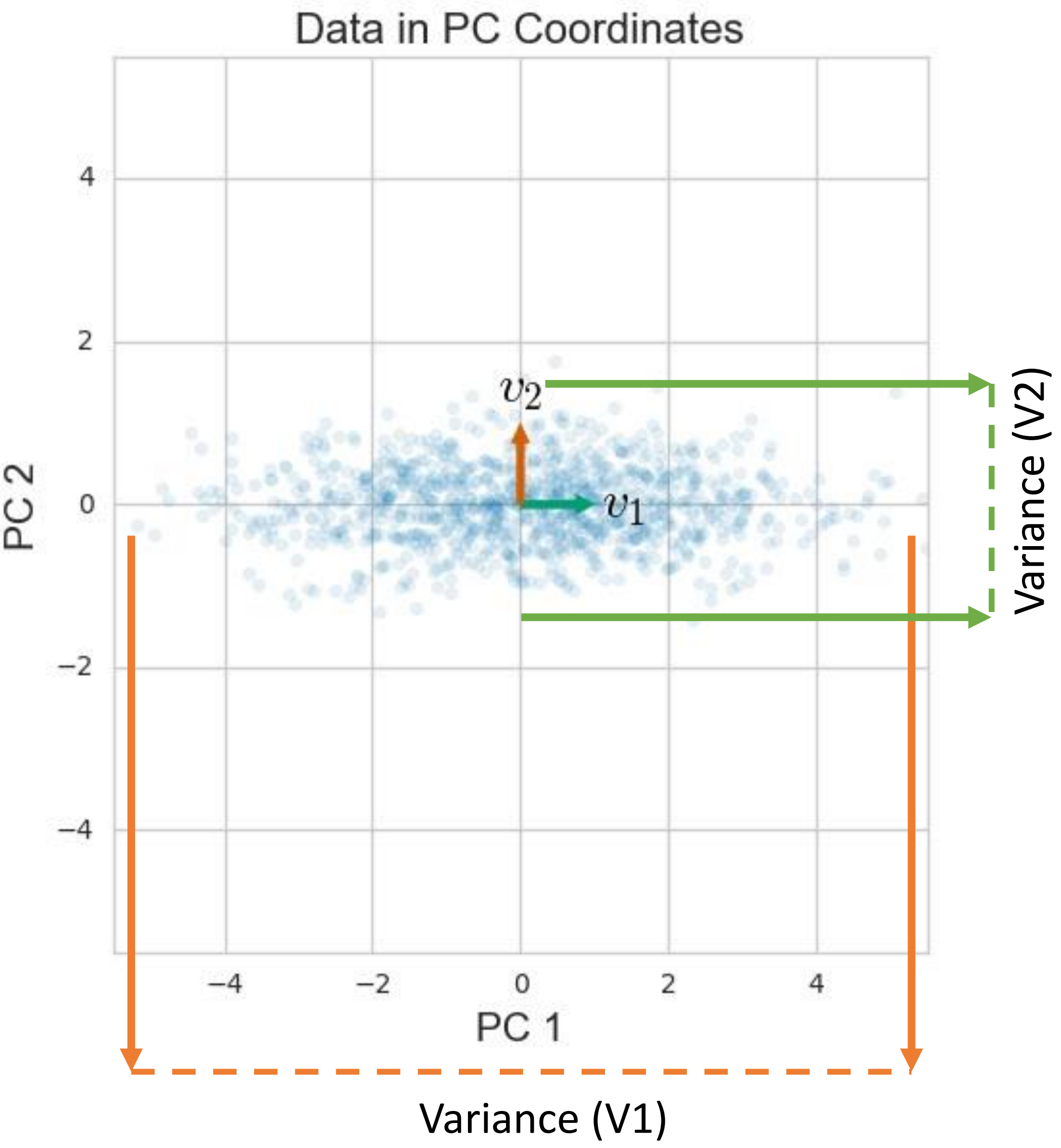


PCA is not used to Predict values. It is a Feature extraction Technique. By PCA we create new features from old (Original) Features but the new features will always be independent of each other.

From the Figure we can see the Variance (V1 and V2) explained by our Original data. PCA is known for Dimension reduction by Increasing Variance so that the Information is fairly retained with very minor loss.

When we have our data in higher dimension space , i.e., more features which can likely affect our model performance or consume too much computational resources that's when PCA comes into picture.

Principal Component Analysis



Now, we have to rotate our axes (X1 and X2) in such a way that they become Principal Components (PC1 and PC2) , we can clearly identify the Transformed data is explaining more Variance (V1).

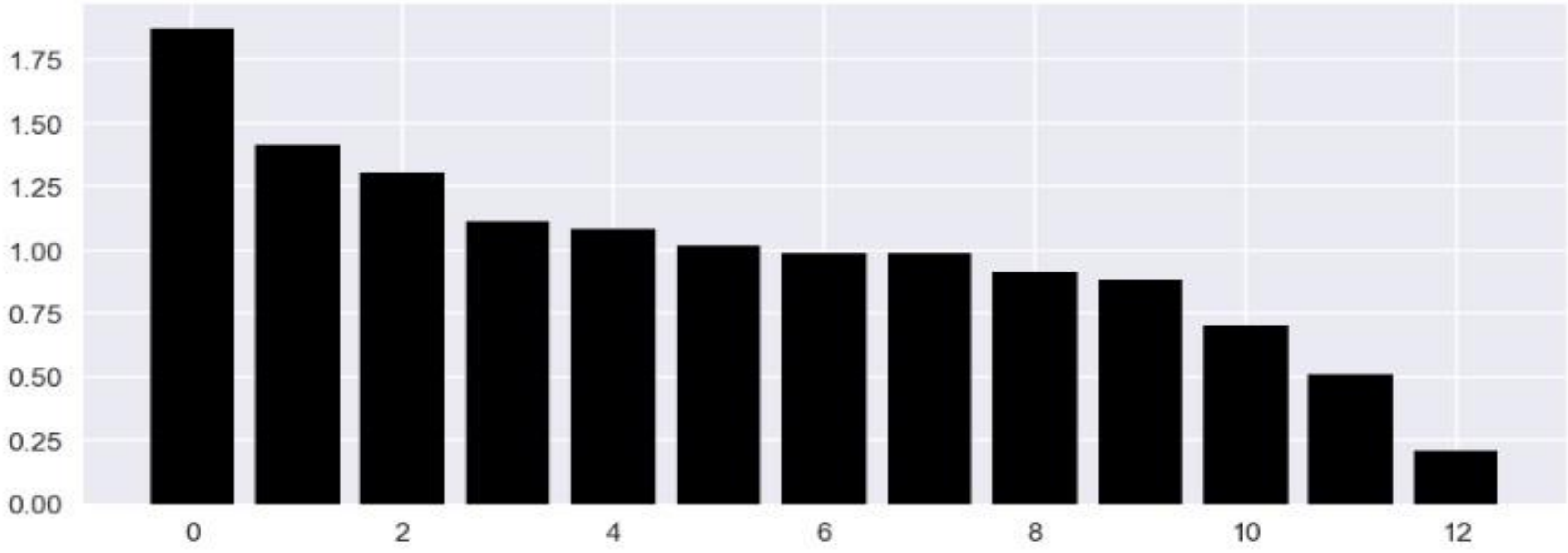
Further we can consider the Transformed data as our independent Variable or Predictor.

Now that we're done, we have to pass our Scaled Dataframe in PCA model and observe the elbow plot to get better idea of explained variance. At 12th component our PCA model seems to go Flat without explaining much of a Variance.

```
In [65]: X = new_df
```

```
In [66]: from sklearn.decomposition import PCA

         pca = PCA(n_components=len(nyc_taxi.columns)-1)
         pca.fit_transform(X)
         var_rat = pca.explained_variance_ratio_
         var_rat

Out[66]: array([0.14409795, 0.10874783, 0.10046737, 0.08575584, 0.08327229,
                0.07812324, 0.07600604, 0.0757774 , 0.07020683, 0.06811033,
                0.05427252, 0.03908354, 0.01607881])
```

```
In [67]: plt.figure(figsize=(15,6))
         plt.bar(np.arange(pca.n_components_), pca.explained_variance_, color="black")

Out[67]: <Container object of 13 artists>
```



Linear Regression

Let's pass the PCA Transformed data in our Machine Learning Regression Algorithms. To begin with , Linear Regression is a good approach, by splitting our Data into Training and Testing (40%). We have to consider **RMSE** as a evaluation Metrics , not ~~R-squared~~. We can also hyper tune our Parameters to minimize the loss (RMSE). We will also calculate **Null RMSE** , which we can set as a benchmark for our Models RMSE.

In [32]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import r2_score, mean_squared_log_error , mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40, random_state=10)
lin_reg = LinearRegression()
model = lin_reg.fit(X_train, y_train)
pred = lin_reg.predict(X_test)
pred
```

Out[32]:
```
array([6.32328104, 6.37618293, 6.49043933, ..., 6.48016468, 6.52877018,
       6.50631427])
```

In [33]:
```
lin_reg.intercept_, lin_reg.coef_
```

Out[33]:
```
(6.4642683980121385,
 array([-1.40745025e-01,  1.79420917e-01, -1.33627040e-02,  8.11089338e-03,
         5.48887552e-02,  1.43327418e-02,  2.61141275e-03,  2.43124232e-02,
        -1.28707489e-02, -4.11236048e-03, -9.05190326e-05,  7.57391559e-03]))
```

Decision Tree and Random Forest

Decision Tree

In [37]:
```
from sklearn.tree import DecisionTreeRegressor

dt = DecisionTreeRegressor(criterion="mse", max_depth=10)
model = dt.fit(X_train, y_train)
pred = dt.predict(X_test)
pred
```

Out[37]:
```
array([6.58052977, 7.51122233, 6.16034433, ..., 6.72750745, 6.36335234,
       7.0337328 ])
```

We've to import Decision Tree Regressor and Random Forest Regressor and imply respective algorithms on our Data and evaluate results.

Random Forest

In [40]:
```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(criterion="mse", n_estimators=5, max_depth=10)
model = rf.fit(X_train, y_train)
pred = rf.predict(X_test)
pred
```

Out[40]:
```
array([6.56425564, 7.51500054, 6.16283568, ..., 6.72435878, 6.36674563,
       7.01395373])
```

--- RMSE Benchmark ---

Null RMSE

```
y_null = np.zeros_like(y_test, dtype=float)
y_null.fill(y_test.mean())
print ("Null RMSE : " + str(np.sqrt(mean_squared_error(y_test, y_null))))
```

```
Null RMSE : 0.7987144307479092
```

Beat me if you can !!

Null RMSE : 0.7987

We've Null RMSE of 0.7987 which is benchmark for our Prediction model's RMSE. Our model's RMSE must be less than equals to Null RMSE (0.7987)

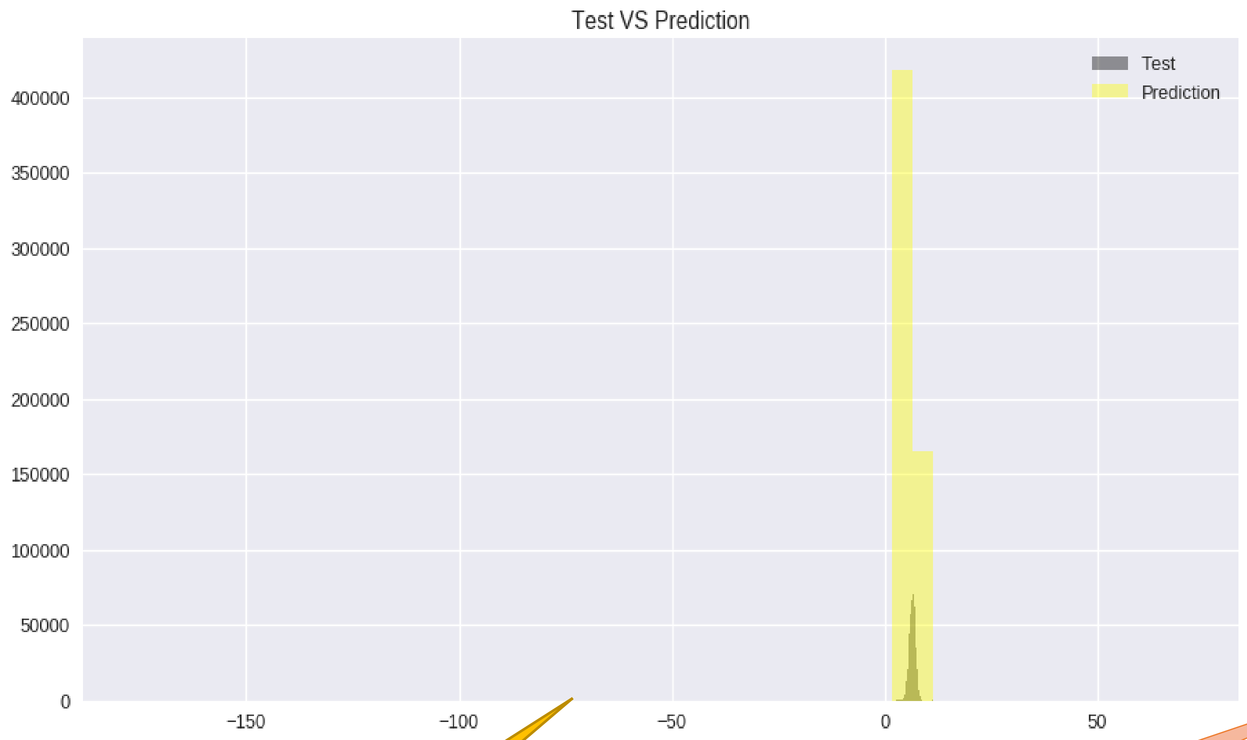
Evaluation Results

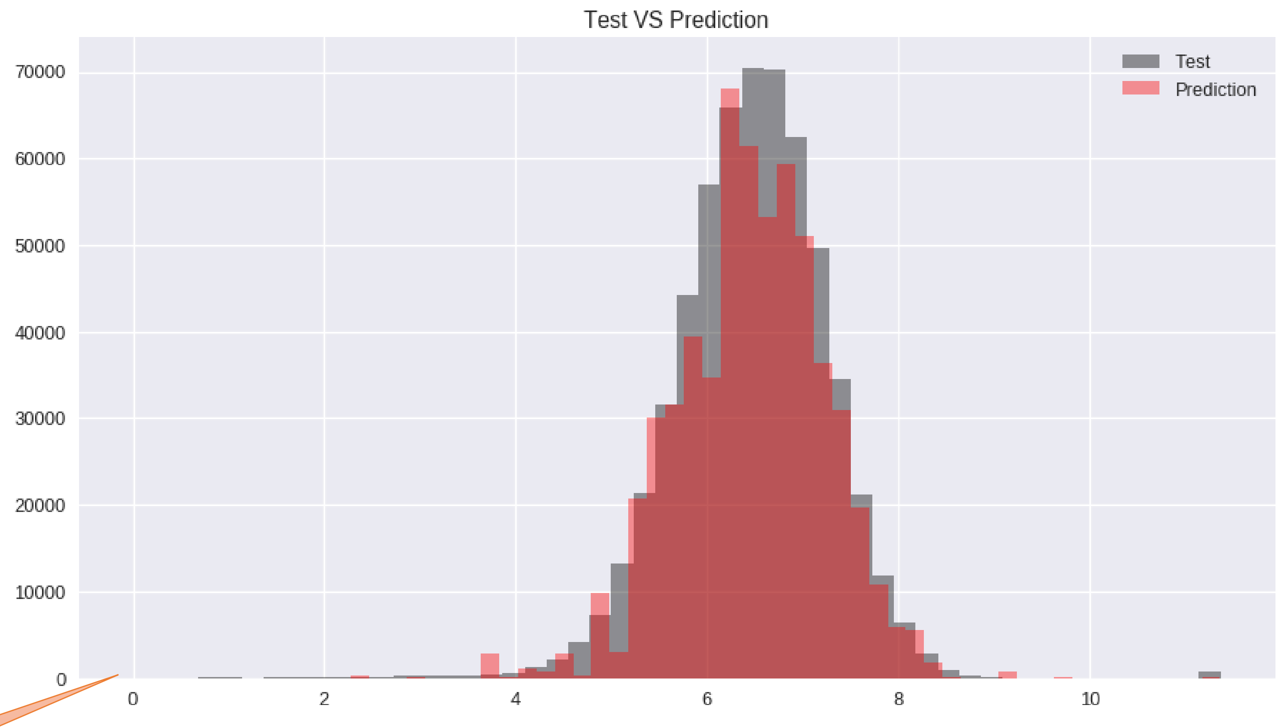
Reference

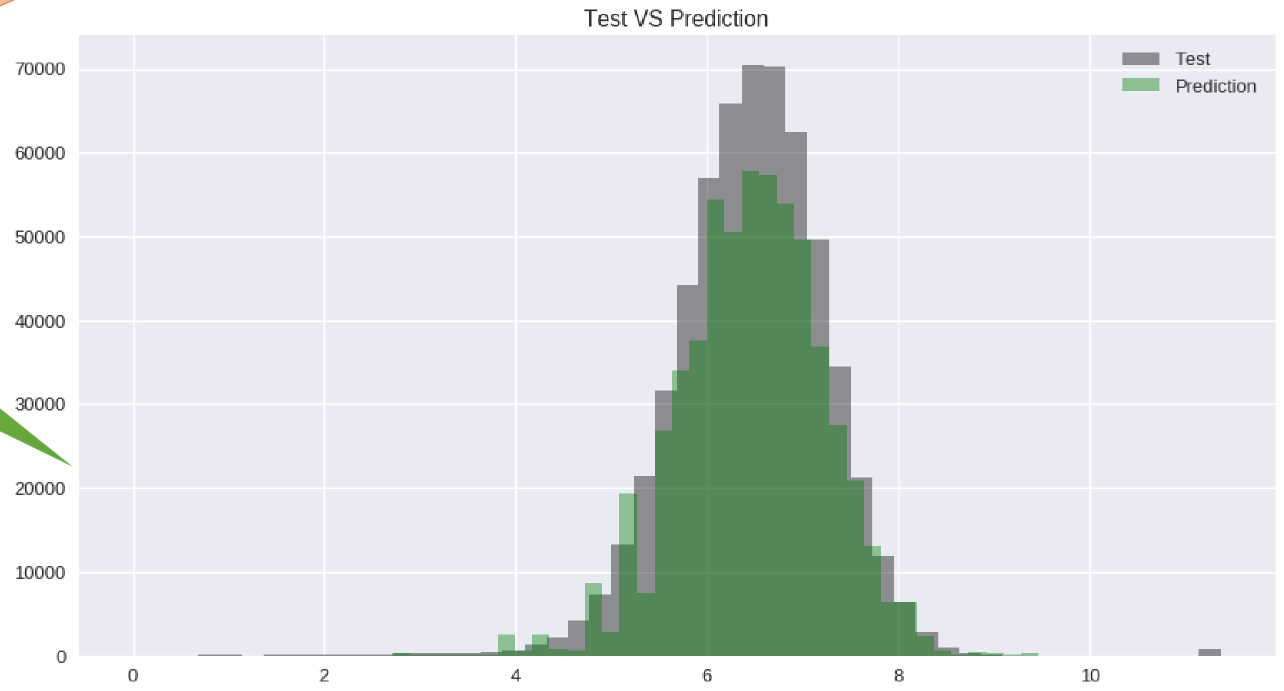
R2-score: Usually must be between 0 and 1, towards 1 considered as good fit.

RMSE: [Value] <= 0.7987 (Null RMSE / Benchmark to Achieve)

Algorithms	Training Score	Cross Validation Score	R2-Score	RMSE
Linear Regression	0.0635	0.0619	-3.5572	0.8455
Decision Tree	0.9238	0.9132	0.9052	0.2364
Random Forest	0.9301	0.9238	0.9145	0.2234







Linear Regression

Decision Tree

Random Forest

Visualizations show us How our models's Predictions are close to Test Data. It is evident that Decision Tree and Random Forest are Performing well.

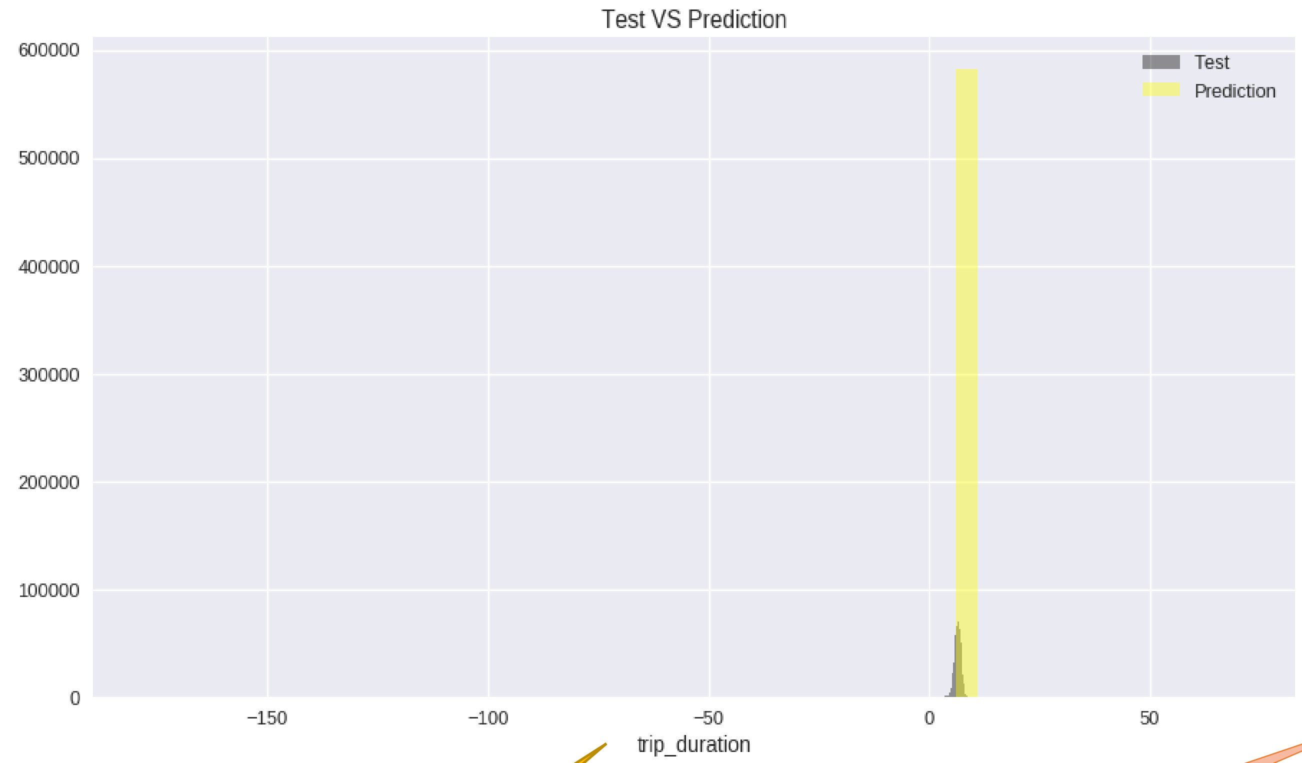
Another Approach…

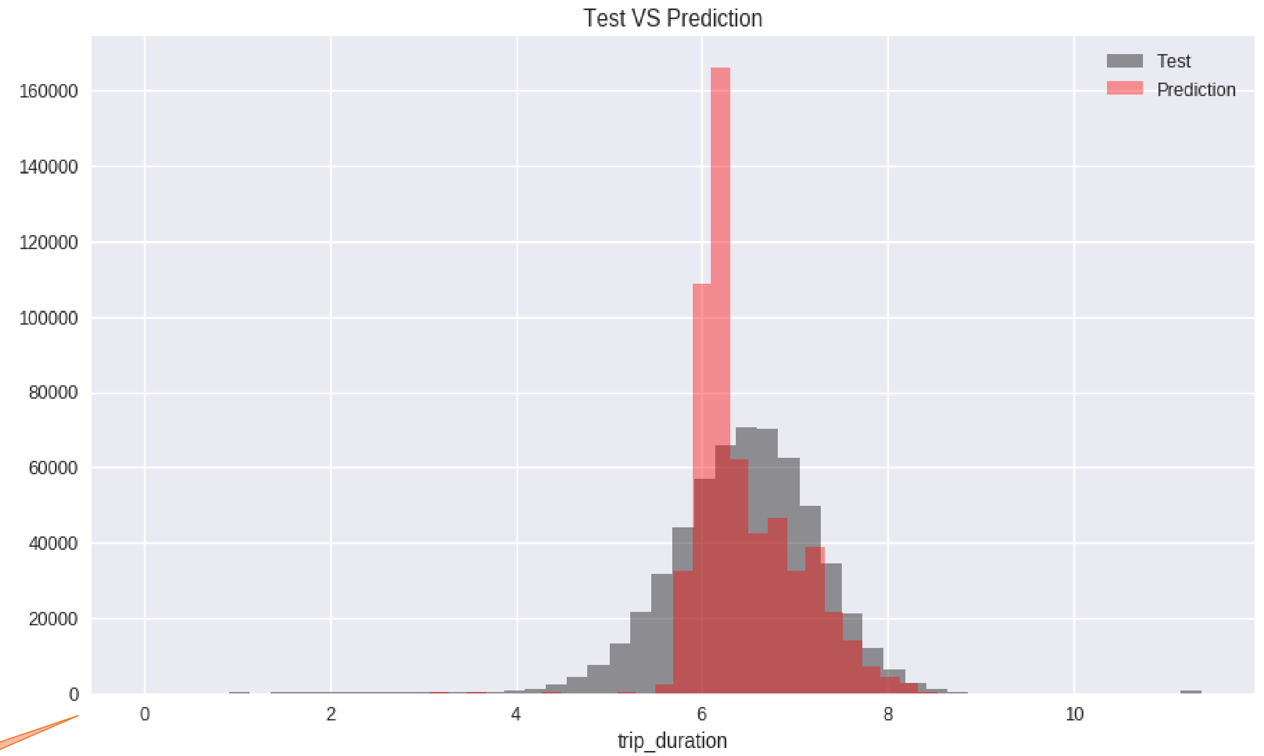
- Another approach we could go with is without PCA, just Standard Scaling Dataset and applying our Algorithms.
- The approach can give us better idea of what works better for us.
- This approach might take great amount of computational resources and time, it will be good if we can run this on Google's Collaboratory, that will eliminate huge computational stress on our system as the program will be running on Server's end and will have their GPU / TPU coming to rescue us.

The "Without PCA" approach..

Remember our Null RMSE: 0.7987

Algorithms	Training Score	Cross Validation Score	R2-Score	RMSE
Linear Regression	0.0649	0.0634	-3.4938	0.8459
Decision Tree	0.4646	0.4569	-0.1677	0.5893
Random Forest	0.4790	0.4680	-0.1823	0.5804

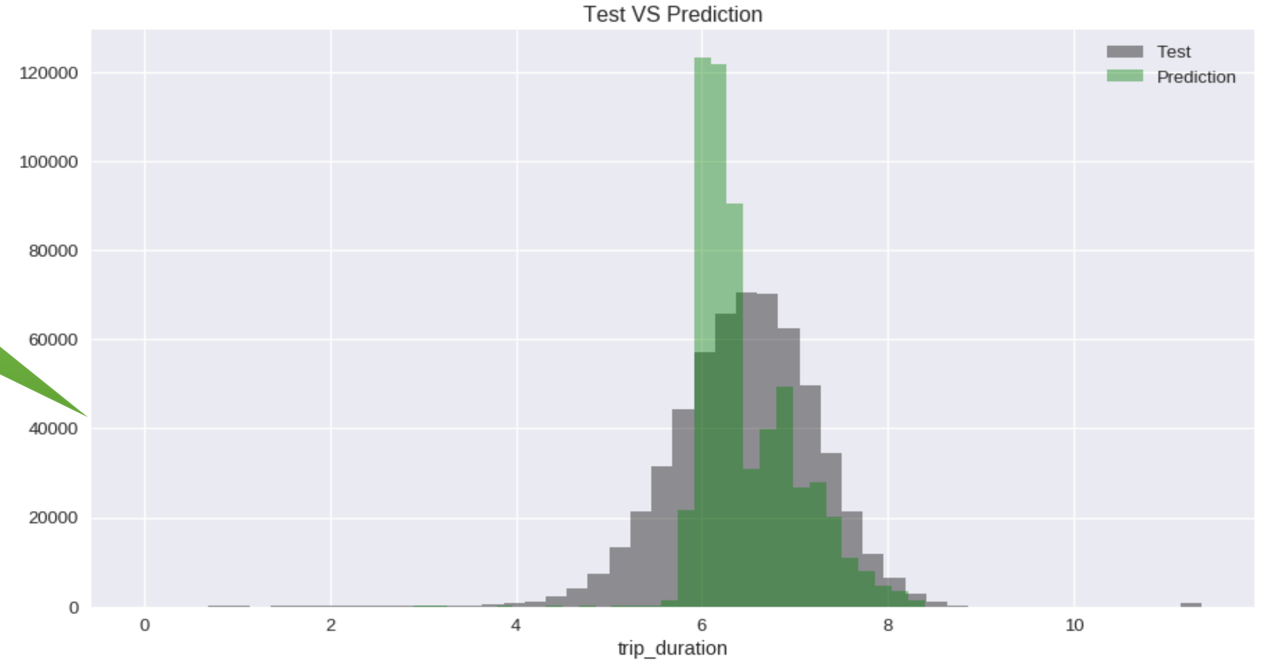




Linear Regression

Decision Tree

Random Forest



Test VS Prediction, How close they are ?

Recommendations

- Apply Standard Scaling on the Dataset to Normalize the values.
- Further, Apply PCA to reduce dimensions, as you'll extract features from our primary DateTime Feature. Those additional features might lead our model to suffer from "Curse of dimensionality" and could drastically affect performance.
- Pass the PCA Transformed data in our ML Regression Algorithms and Evaluate results.

Insights

- ✓ Observed which taxi service provider is most Frequently used by New Yorkers.
- ✓ Found out few trips which were going from 528 Hours to 972 Hours, possibly Outliers.
- ✓ With the help of Tableau, we're able to make good use of Geographical Data provided in the Dataset to figure figure prominent Locations of Taxi's pickup / dropoff points.
- ✓ Also, found out some Trips of which pickups dropoff point ended up somewhere in North Atlantic Sea.
- ✓ Passenger count Analysis showed us that there were few trips with Zero Passengers.
- ✓ Monthly trip analysis gives us a insight of Month – March and April marking the highest number of Trips while January marking lowest, possibly due to Snowfall.
- ✓ In a day, we could observe that 4pm to 8pm is the time when New Yorkers Rush too much.
- ✓ Observations says that Friday's and Saturday's are those days in a week when New Yorkers prefer to get out of their home.

If, in New York, you arrive late for an appointment, say, "I took a taxi".

— *Andre Maurois* —

[redacted]