CAB FARE PREDICTION PROJECT REPORT

NAME: NIRAV THANKI

Project Name: Cab Fare Prediction

❖ Problem Statement:

You are a cab rental start-up company. You have successfully run the pilot project and now want to launch your cab service across the country. You have collected the historical data from your pilot project and now have a requirement to apply analytics for fare prediction. You need to design a system that predicts the fare amount for a cab ride in the city.

❖ The Data:

Description of Train & Test Dataset which provided for this problem:

Feature Name	Feature Description	Feature Data Type		Feature Type
		Train Data	Test Data	
fare_amount	Cost of cab ride. This is the value to be predicted. It's not present in Test data	Python: object	NA	Target Variable
		R: Factor	NA	
pickup_datetime	Date & Time when trip started	Python: object	Python: object	Predictor Variable
		R: Factor	R: Factor	
pickup_longitude	Longitude coordinate of where trip started	Python: float64	Python: float64	Predictor Variable
		R: numeric	R:numeric	
pickup_latitude	Latitude coordinate of where trip started	Python: float64	Python: float64	Predictor Variable
		R:numeric	R:numeric	
dropoff_longitude	Longitude coordinate of where trip ended	Python: float64	Python: float64	Predictor Variable
		R:numeric	R:numeric	
dropoff_latitude	Latitude coordinate of where trip ended	Python: float64	Python: float64	Predictor Variable
		R: numeric	R: numeric	
passenger_count	Number of passenger in cab ride	Python: float64	Python: int64	Predictor Variable
		R: numeric	R: integer	

Length of Train Data set is 16067 rows and 7 variables in which 'fare_amount' is target variable and remaining all are Predictor variables.

Length of Test Data set is 9914 rows and 6 features.

❖ Hypothesis Generation:

- **Trip Distance:** Distance should be directly affect far amount. If the distance to be travelled is more, then fare should be higher.
- **Time of Travel:** During the pick hour, the taxi fare may be higher because there is too much demand and not enough cab to service the additional demand.
- **Day of Travel:** Fare amount may be different on weekday and weekends.
- **Airport Trip:** Most of time airport trips have fixed fare.

- **Pickup or Drop-off closeness:** Fare amount may different on kind of closeness between pickup and drop-off.

❖ Exploratory Data Analysis:

In this section I will clean the dataset and understand relationship between features and used this understanding to create better features.

Missing Value:

First, let's check is there any missing value present in Train & Test Dataset?

Feature Name	Train Data	Test Data
	No. of Missing value	No. of Missing value
passenger_count	55	0
fare_amount	24	0
pickup_datetime	0	0
pickup_longitude	0	0
pickup_latitude	0	0
dropoff_longitude	0	0
dropoff_latitude	0	0
Total	**79**	**0**
% of Missing Value	**0.49%**	**0.00%**

Account of missing value is less than 0.5% of Train Dataset. so it's advisable to remove rows which contain missing value. Still I have enough observations to move ahead.

Train Data set contain **15988** rows after removing Missing values. Means I have excluded **79** rows.

Distribution of Fare Amount (Target Variable):

Feature cleaning is required first because as checked there is some string present in 'fare_amount'.

After cleaning I have changed feature type from "object/Factor" to "float64/numeric" in Python and R respectively.

Remove one row which contain string. So, now our Train data contain **15987** rows.

Summary statistic of fare amount:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-3	6	8.5	15.03	12.5	54343

Some insight of fare amount.

- There are 3 observations where fare amount is less than 0
- There are 1 observation where fare amount is equal to 0

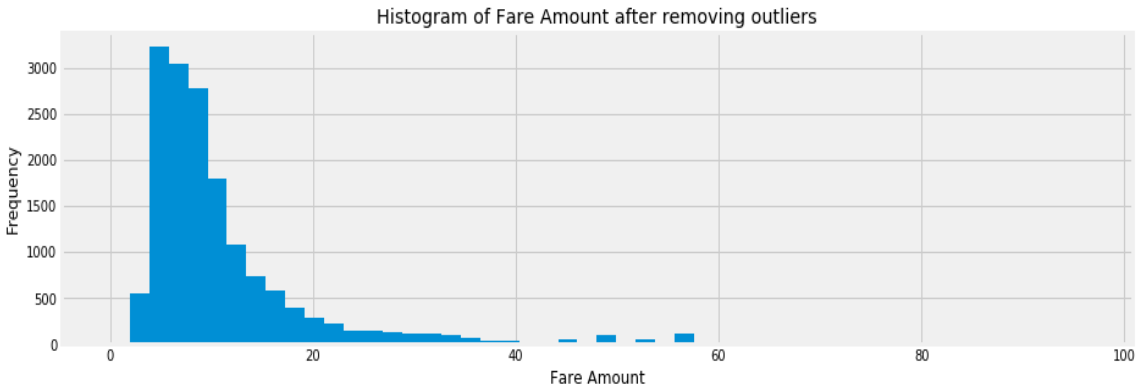
- There are 6 observations where fare amount is less than 2.5
- There are 9 observations where fare amount is greater than 100
- There are 2 observations where fare amount is greater than 500

This project is based on New York City Taxi Fare Prediction. So as per nyc.gov website Minimum fare charges is 2.5 and as per above insights there are 9 observations in which fare charged greater than 100. So I can say it's outliers and where ever fare charges less than 2.5 it's inliers.

Hence I have set range of fare amount as per below

fare amount should be greater than or equal to 2.5 & Less than or equal to 100.

Histogram of Fare amount:



Train data contain **15972** rows after removing inliers and outliers from fare amount. Means I have excluded total **15** rows.

Distribution of Location Related Variables:

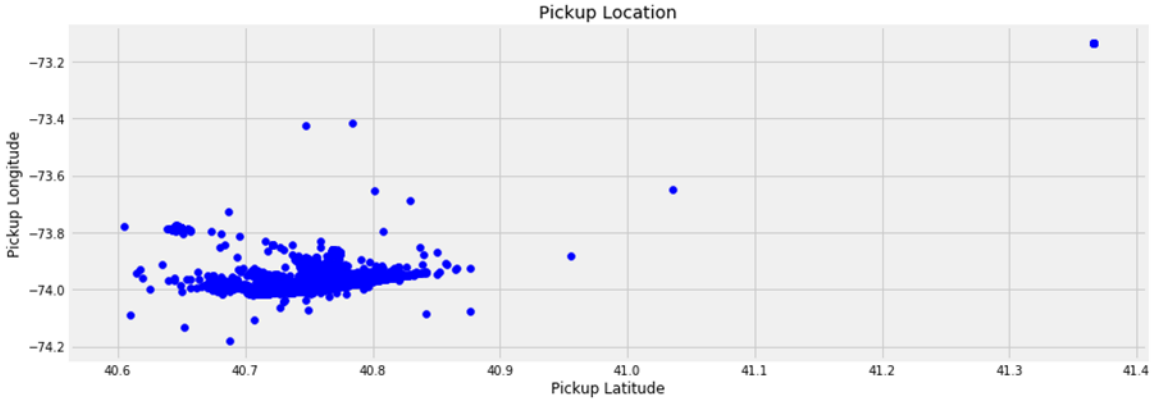
Minimum & Maximum summary of Location Related Variables of Train & Test Data:

Feature Name	Train Data		Test Data	
	min	max	min	max
pickup_longitude	-74.4382	40.76613	-74.2522	-72.9865
dropoff_longitude	-74.4293	40.80244	-74.2632	-72.991
pickup_latitude	-74.0069	401.0833	40.57314	41.70956
dropoff_latitude	-74.0064	41.36614	40.56897	41.69668

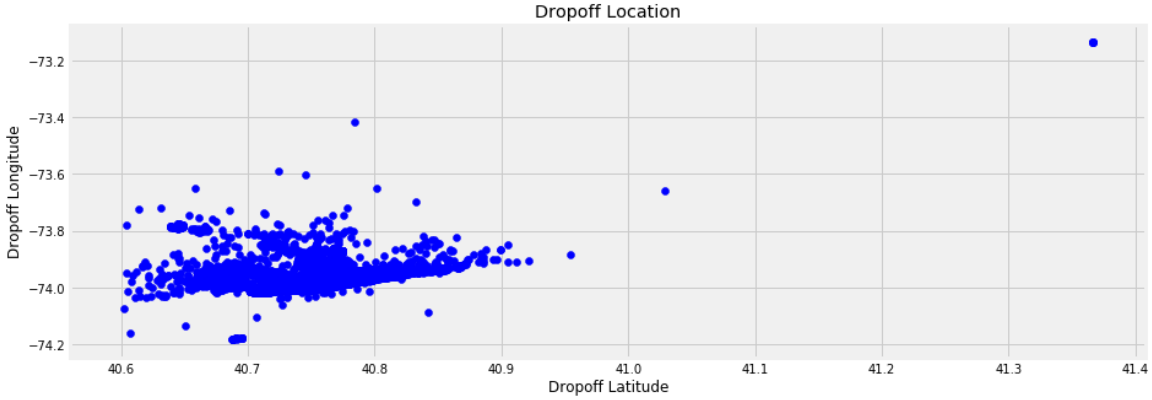
As per above Min-Max summary range of latitude & longitude are between 40.6 to 41.8 and -74.3 to -73.0 respectively in Test dataset. Hence without digging further I have set range of latitude & longitude as per Test dataset to remove anomalies.

Train data contain **15610** rows after removing anomalies from Location related variables. Means I have removed **362** rows.

Distribution of Pickup Location:



Distribution of Drop-off Location:



Distribution of Passenger Count:

Feature cleaning is required here because passenger count should be an Integer hence I have changed feature type from “float64/numeric” to “int64/integer” in Python and R respectively.

Minimum & Maximum summary of Passenger count:

Passenger Count		
	min	max
Train Data	0	5345
Test Data	1	6

Without any domain knowledge I can easily say that there are some anomalies present in passenger count.

Some insight of passenger counts in Train Data:

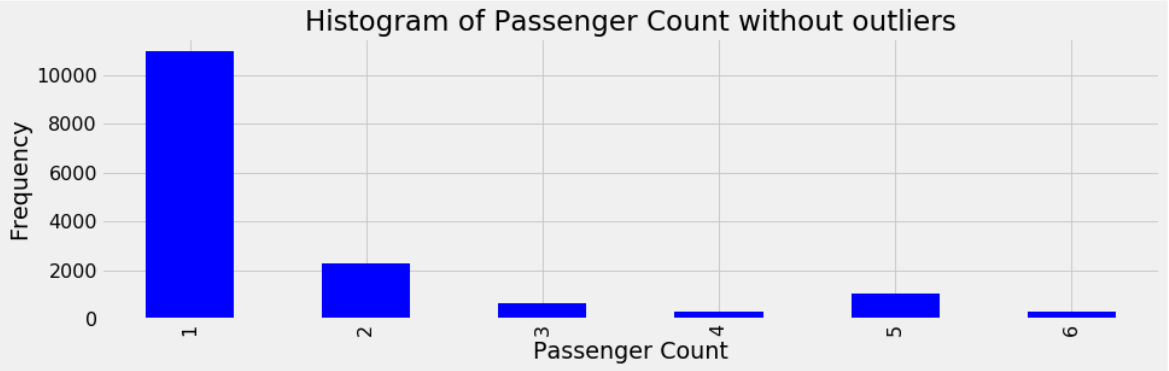
- There are 56 observations where Passenger count is equal to 0.
- There are 15299 observations where Passenger count is Less than 6.
- There are 294 observations where Passenger count is equal to 6.
- There are 17 observations where Passenger count is Greater than 6.

As per Test Dataset I have set limit of Passenger count between 1 to 6.

Train data contain **15537** rows after removing anomalies form passenger count. Means I have excluded **73** rows from Train Dataset.

Correlation between passenger count & target variable (fare amount) is 0.0068. it's means that Passenger Count has no strong linear relationship with fare amount.

Histogram of passenger count:



As per above histogram approx. 11000 trips contain only 1 passenger and greater than 2000 trips contain 2 passengers.

Pickup Date & Time:

Feature cleaning Is required here because some observation Is not in format. Before moving ahead, I have checked feature type of "pickup_datetime" It was "Object/Factor" in Python & R respectively.

So it's required to convert it into proper type. Hence I have converted 'pickup_datetime' into "datetime64[ns, UTC]/POSIXt" in Python & R respectively.

After conversation I have checked is there any "NA" occurred or not? And I found that after conversation there is one "NA" is occurred hence I removed it.

Train data contain **15592** rows after removing NAN value form pickup date time. Means I have excluded **1** row from Train dataset.

Detail of removed rows in each steps while doing Exploratory Data Analysis:

Train Data	
Reasons	**No. of Rows removed**
Due to Missing Value	79
Due to Outliers in Fare Amount	15
Due to Outliers in Location related variables	362
Due to Outliers in Passenger Count	73
Due to Missing value in Pickup date & time	1
Total No of rows removed	**530**

Initially length of Train data set was **16067** and after removing rows due to anomalies & Missing values now length of Train data set is **15537** means I have lost approx. **3.3%** of data.

Range of Features which is set during Exploratory Data Analysis:

Feature Name	Range
fare_amount	2.5 - 100
pickup_longitude	-74.3 to -73.0
pickup_latitude	40.6 to 41.8
dropoff_longitude	-74.3 to -73.0
dropoff_latitude	40.6 to 41.8
passenger_count	1 to 6

❖ Feature Engineering:

Feature engineering is the process of creating new features – predictor variables out of an existing dataset. Because a machine learning model can only learn from given features and properly constructing features will determine how well my model performs.

Create New Features 'abs_lat_diff' & 'abs_lon_diff':

I know that the fare of a cab ride is proportional to the distance, I want to use the start and stop points to try and find the distance travelled. One rough approximation of distance is the absolute difference between the start & end latitudes and longitudes.

Correlation of created features with target variable:

abs_lat_diff	abs_lon_diff
0.6	0.74

These features give me is a relative measure of distance because they are calculated in terms of latitude and longitude and not an actual metric. These features are useful for comparison

Create New Feature 'distance' (in KM):

To calculate a more realistic distance between the pickup and drop-off, I used the Haversine distance. This is the Great Circle distance, representing the shortest distance along the surface of the Earth connecting two points taking into account the fact that the Earth is a sphere. It's not the best measure because the cabs do not travel along lines, but it's more accurate in terms of an absolute distance.

The formula for Haversine distance is:

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

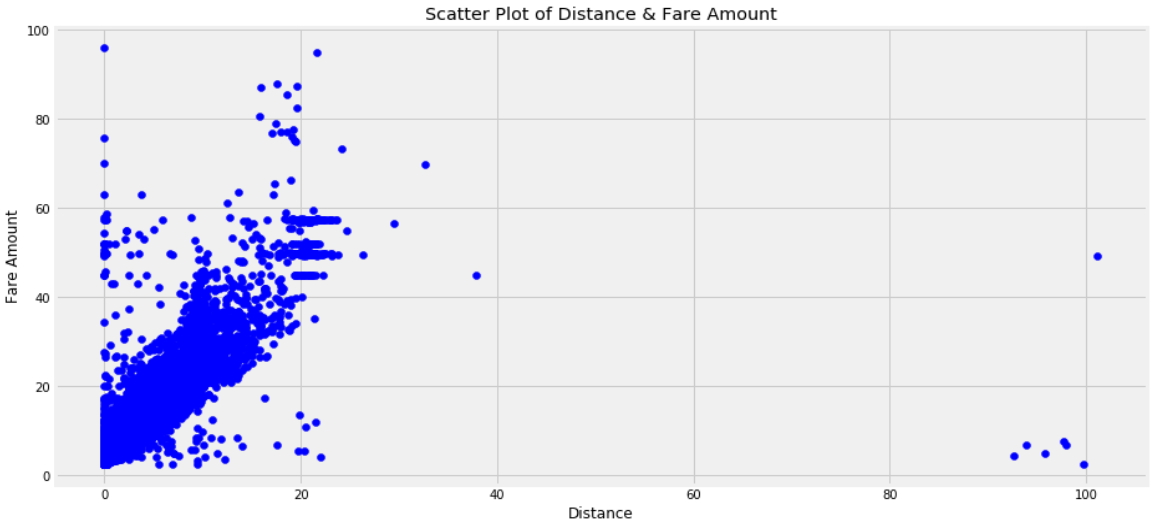
Where,

R = earth's radius (mean radius = 6371 KM)

My thanks go to this Stack Overflow answer:

https://stackoverflow.com/a/29546836

Scatter Plot of Distance & Fare Amount:



A scatter plot between trip distance and fare amount showed that, there is a linear relationship. The larger haversine distances tend to have larger fares as expected.

Correlation between distance & fare amount (Target Variable) is 0.76

Summary statistic of distance of Train & Test Data set.

	Min	1st Qu.	Median	Mean	3rd Qu.	Max.
Train Data	0.000	1.257	2.168	3.357	3.888	101.095
Test Data	0.000	1.298	2.217	3.435	4.045	99.996

As digging further just for curiosity there are 155 rows of Train data & 85 rows of Test data where distance travelled is '0' and still fare is charged on those trips.

In Train data surprised thing is that out of 155 rows there are 140 rows where fare has been charged greater than 2.5.

Create Date & Time Related Features:

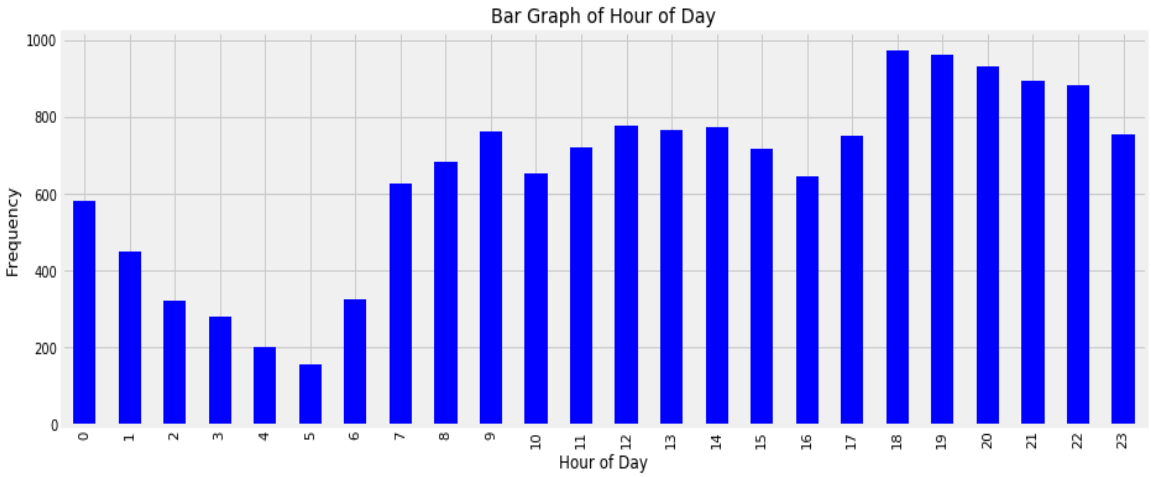
In exploratory data analysis I have converted 'pickup_datetime' variable in proper type so, now it' time to extract useful insight form it.

With the help of 'Pandas' & 'dplyr' in Python & R respectively I have extract below features.

- Hour_of_day
- Month
- Year
- Day
- Weekday

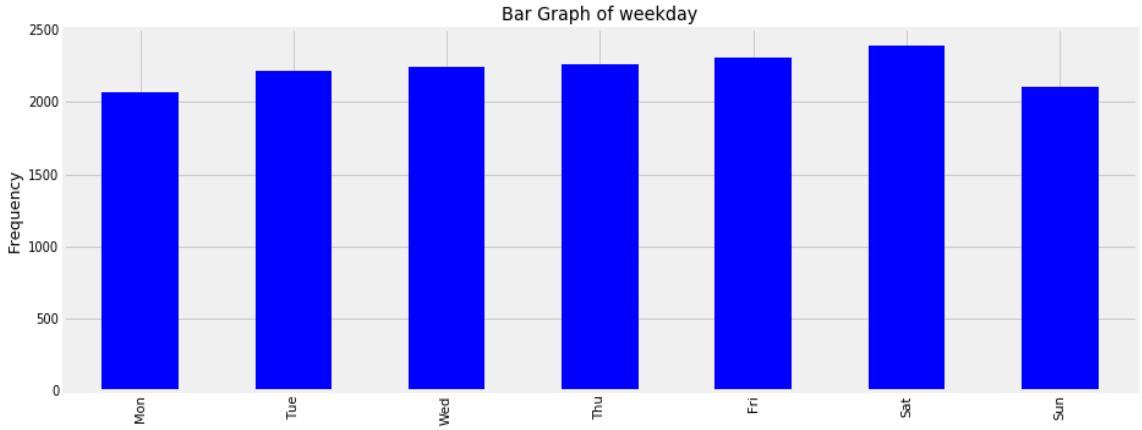
Hour of Day:

Bar plot of hour of day:



Hour of day is directly affect the cab rides. Frequency of rides between 18:00 to 22:00 is greater than 800 & on the other hand between 1:00 to 4:00 less than 400.

Bar plot of day of Travel:



As per above bar plot I can say that Friday and Saturday are highest frequency of cab rides as compare to other days.

Create Airport Distance related features:

Some trips, like to/from an airport, are fixed fee

Another way to explore this data is to check trips to/from well-known places. E.g. a trip to JFK airport. Depending on the distance, a trip to an airport is often a fixed price. Hence due to curiosity I have tried to add some location coordinators of airport and checked their correlation with my target variable (fare amount).

Correlation of added airport distance with fare amount (target variable):

Feature Name	jfk_dist	ewr_dist	lga_dist	sol_dist	nyc_dist
Correlation with Target	-0.39	0.32	0.06	0.28	0.33

After exploratory data analysis and feature engineering we have below feature for Train & Test data:

No.	Feature Name	Feature Type	
		Python	R
1	fare_amount	float64	numeric
2	pickup_datetime	datetime64[ns, UTC]	POSIXCt
3	pickup_longitude	float64	numeric
4	pickup_latitude	float64	numeric
5	dropoff_longitude	float64	numeric
6	dropoff_latitude	float64	numeric
7	passenger_count	int64	integer
8	abs_lat_diff	float64	numeric
9	abs_lon_diff	float64	numeric
10	distance	float64	numeric
11	hour_of_day	int64	numeric
12	month	int64	numeric
13	year	int64	numeric
14	day	int64	numeric
15	weekday	int64	numeric
16	jfk_dist	float64	numeric
17	ewr_dist	float64	numeric
18	lga_dist	float64	numeric
19	sol_dist	float64	numeric
20	nyc_dist	float64	numeric

❖ Split Train Dataset into Train & Valid:

In order to evaluate the performance of model, I split the data into a training set (80% of Train Dataset) and validation set (20% of Train Dataset)

❖ Model Evaluation:

There are number of error metrics are present to evaluate the performance of Regression Model.

I chose RMSE to evaluate our prediction because it favours consistency and heavily penalizes predictions with a high deviation from the true number.

Root Mean Square Error (RMSE) seems to be a perfect match as it is similar to the Mean Absolute Error (MAE, which gives an average of all the errors without considering their direction), but it amplifies the magnitude of large errors. MSE has similar characteristics to RMSE, but is not as human friendly, as the values are on a different scale and thus not directly comparable and on other hand RMSE is just the square root of MSE. The square root is introduced to make scale of the errors to be the same as the scale of targets.

Taking the square root of the average squared errors has some interesting implications for RMSE. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors.

RMSE of 4.0 would give me an idea that most results are within 3-4 of the true price. This means the RMSE should be more useful because large errors in cab fare are particularly undesirable.

Calculation of RMSE:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Where,

Yj = Actual expected Output

Y^j = Model's Prediction

I have also calculated Mean Absolute Percentage Error (MAPE) for Train & Valid Dataset.

The mean absolute percentage error (MAPE) is the percentage equivalent of MAE. but with adjustments to convert everything into percentages.

Just as MAE is the average magnitude of error produced by my model, the MAPE is how far the model's predictions are off from their corresponding outputs on average. Like MAE, MAPE also has a clear interpretation since percentages are easier for people to conceptualize.

MAPE's weaknesses actually stem from use division operation. Now that I have to scale everything by the actual value, MAPE is undefined for data points where the value is 0. Similarly, the MAPE can grow unexpectedly large if the actual values are exceptionally small themselves. Finally, the MAPE is biased towards predictions that are systematically less than the actual values themselves. That is to say, MAPE will be lower when the prediction is lower than the actual compared to a prediction that is higher by the same amount.

Calculation of MAPE:

$$\mathrm{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Where,

Yi = Actual expected Output

Y^j = Model's Prediction

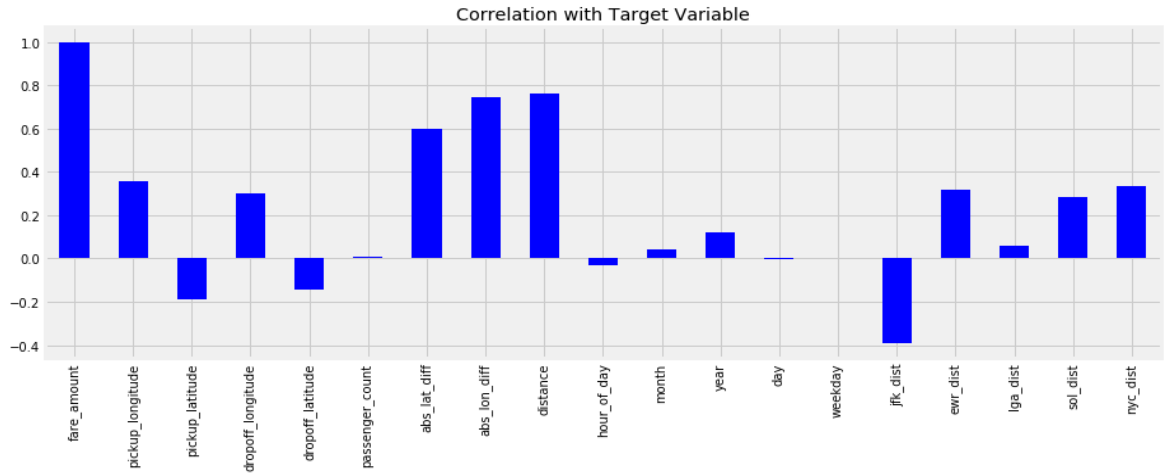
❖ Machine Learning:

After cleaned my data, removed any outliers, removed any errors, and featured engineered my pickup location and drop-off location to get the distance between the two points, it's time to implement my data into a Machine Learning Model. As proven by the "No free lunch" theorem, there is no algorithm that is always superior to all others.

I used three types of Machine Learning Models. I used a Multiple Linear Regression, a Random Forest Regression, and a Gradient Boosting Regression.

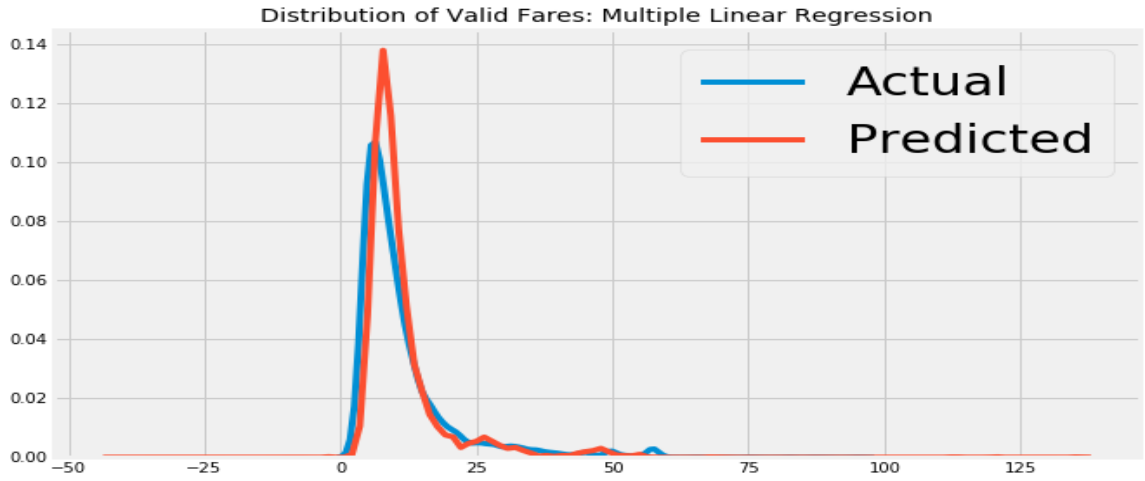
Multiple Linear Regression:

First choice, of model for establishing a baseline on a regression task is a Simple/Multiple Linear Regression. The Multiple Linear Regression model allows me to exploit linear patterns in the data set. Moreover, if I look at the Pearson correlation of the features with the fare amount for this problem, I find several very strong linear relationships with the help of below Correlation plot.



Based on the strength of the linear relationships between some of the features and the target, I can expect a linear model to do reasonably well. The Result of Linear Regression shows below.

Result of Multiple Linear Regression (with all variables):



I select below mention 11 features which p-value is less than 0.05 & still get approx. same result.

pickup_longitude	pickup_latitude	Month	Year
Distance	jfk_dist	ewr_dist	sol_dist
nyc_dist	abs_lat_diff	abs_lon_diff	-

Result of Multiple Linear Regression (with less than 0.05 p-value variables):

Multiple Linear Regression is not very flexible, and has a very high bias. Linear Regression is also highly susceptible to outliers as it tries to minimize the sum of squared errors. there is a limit to its performance. At a certain point, adding more training on more data does not improve predictions.

Random Forest:

The Random Forest is a more flexible model than The Linear Regression which means it has a reduced bias, it can fit the training data better. The random forest also generally has low variance meaning it can generalize to new data. For this problem, The Random Forest outperforms the Linear Regression

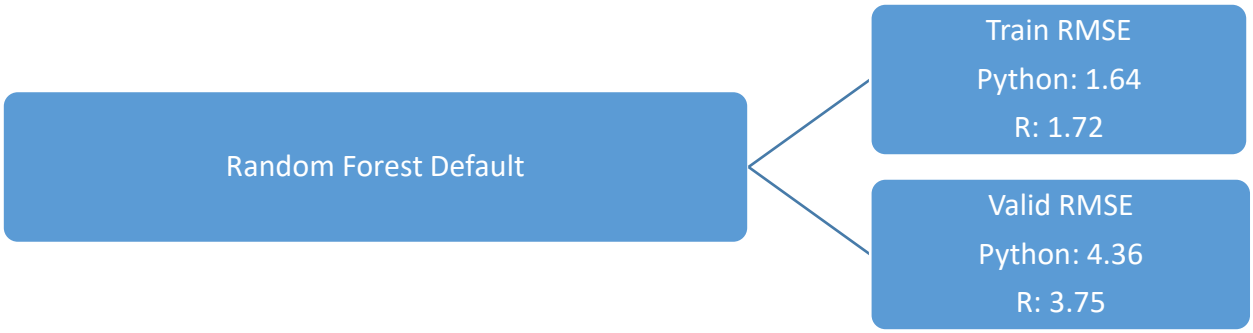
The reason a Random Forest typically outperforms a Linear Regression is because it has more flexibility, lower bias, and it has reduced variance because it combines together the predictions of many decision trees. A Linear Regression is a simple method and as such has a high bias, it assumes the data is linear. A linear regression can also be highly influenced by outliers because it solves for the fit with the lowest sum of squared errors.

Random Forest is far more flexible than a Linear Regression model. This means lower bias, and it can fit the data better.

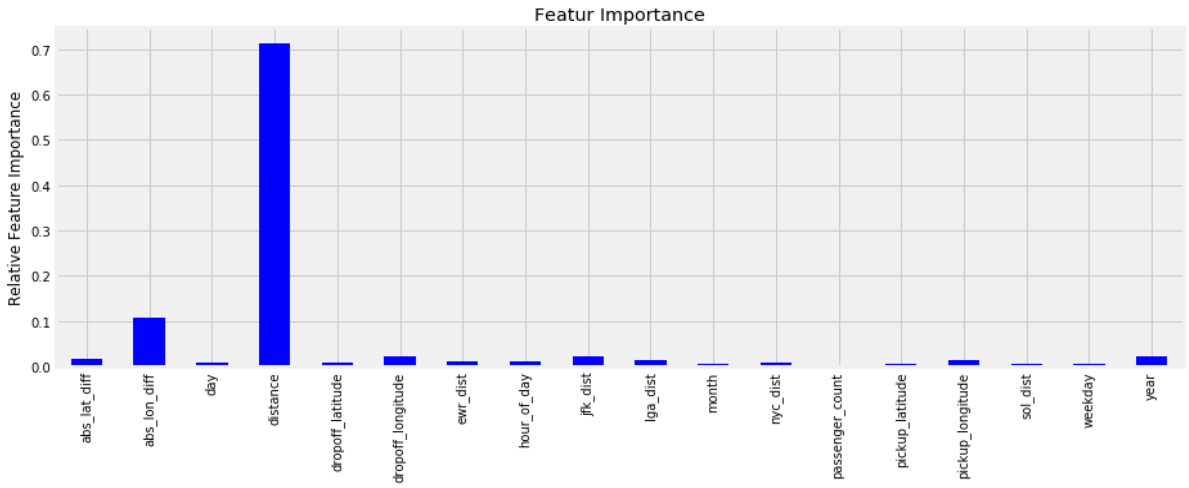
Random Forest uses Bagging method for predictions. In Random Forest, multiple decision trees are created and in Bagging, it takes bootstrap samples (with replacement) of data set and each sample trains a weak learner and the output is the average prediction by each of these trees.

The tree regression model is capable of representing complex decision boundaries, I chose Random Forest because it prevents overfitting and robust against outliers.

Result of Random Forest Default:

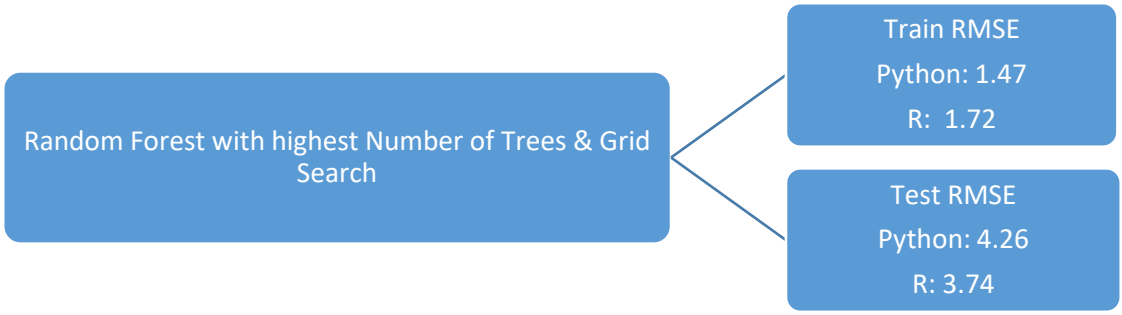


Relative Feature Importance as per Random Forest Default:

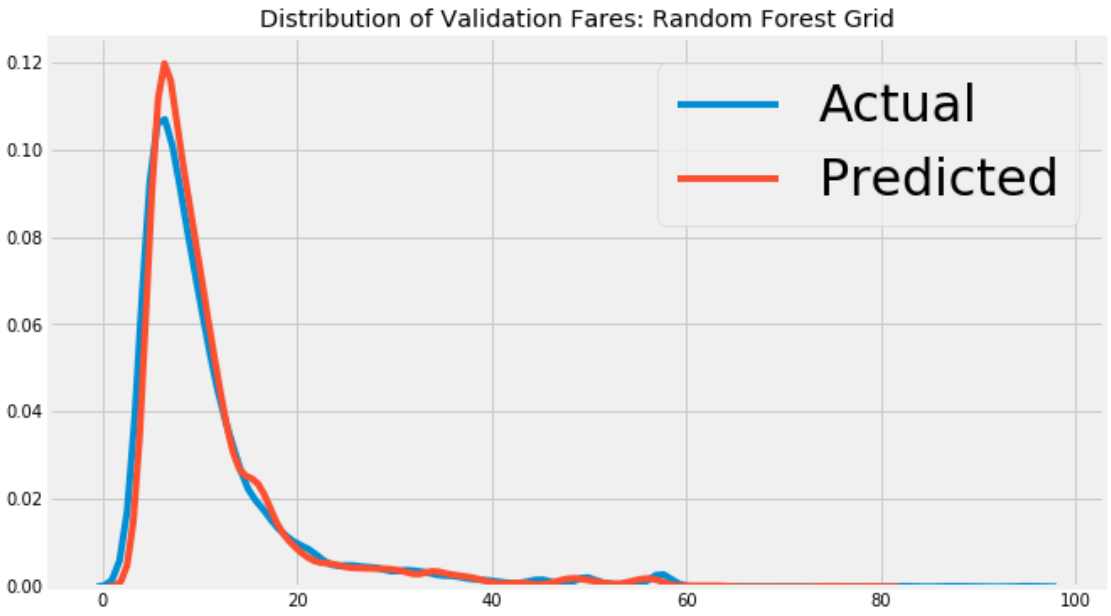


I have increased Number of Trees in Random Forest & also tried grid search option for getting lowest RMSE score as well prevent overfitting.

Result of Random Forest with increase Number of Trees:



After increasing Number of trees in Random Forest my RMSE scores slightly decrease for Train & Valid Dataset.



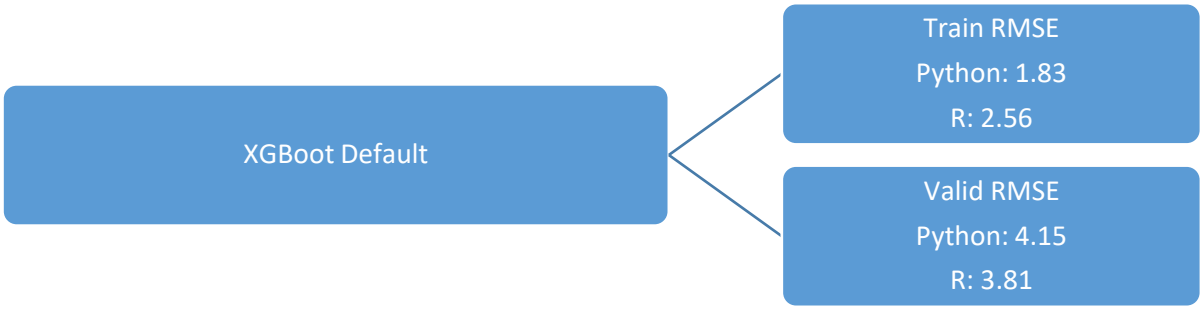
XGBoost:

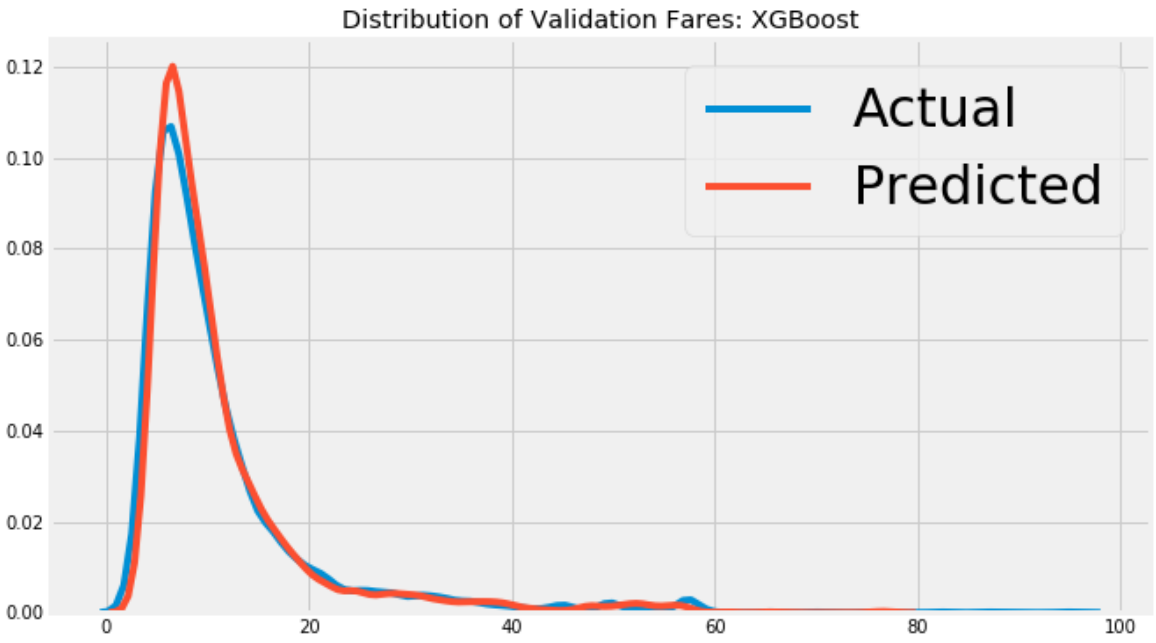
There are still steps to take that can improve the model. My next approach to try an even more complex model, such as a gradient boosting machine.

XGBoost is an advanced gradient boosting algorithm. It is a highly sophisticated algorithm, powerful enough to deal with all sorts of irregularities of data. XGBoost is a boosting tree based algorithm. In this method, multiple weak learners are ensemble to create strong learners. Boosting uses all data to train each learner. But instances that were misclassified by the previous learners are given more weight, so that subsequent learners can give more focus to them during training.

XGBoost is very fast, takes quite less RAM to run, and focuses on the accuracy of the result.

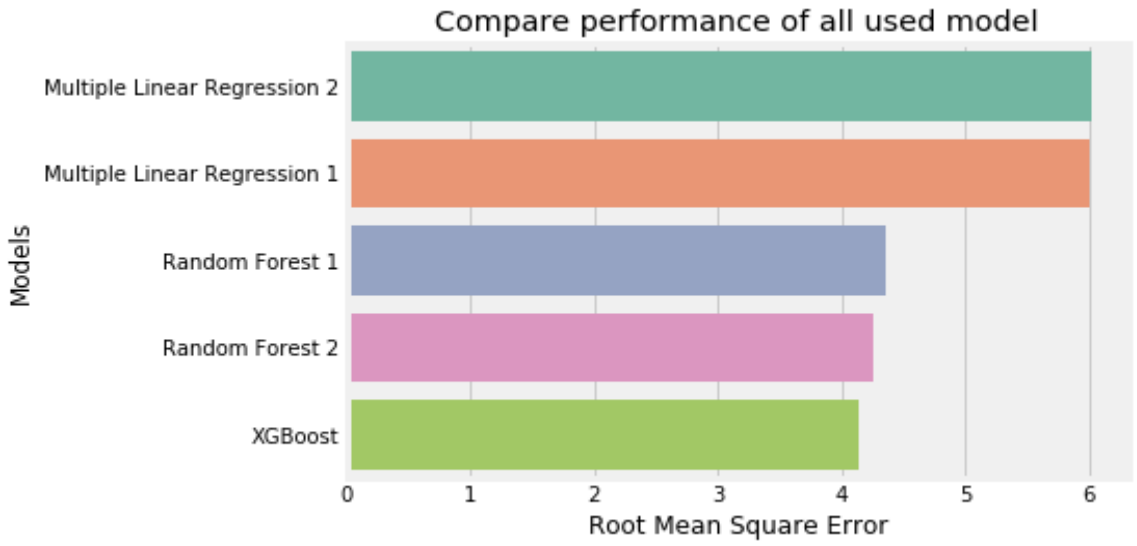
Result of XGBoot:



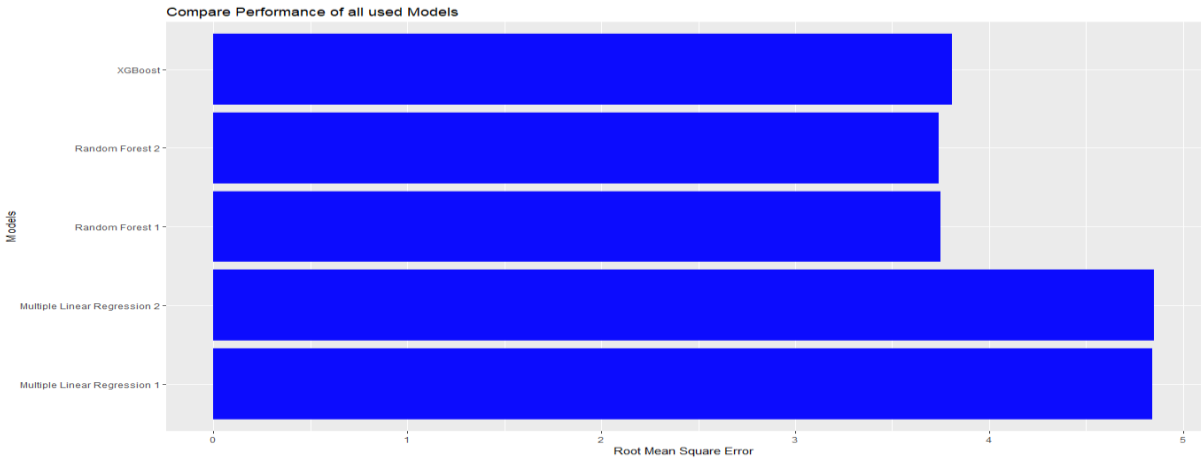


Comparison of Performance of all the used Models:

Result of Python:



Result of R:



Result of Python		
Model	**Train RMSE**	**Valid RMSE**
Multiple Linear Regression 1	4.76	6.01
Multiple Linear Regression 2	4.77	6.02
Random Forest 1	1.64	4.36
Random Forest 2	1.47	4.26
XGBoost	1.83	4.15

Result of R		
Model	**Train RMSE**	**Valid RMSE**
Multiple Linear Regression 1	5.04	4.84
Multiple Linear Regression 2	5.04	4.85
Random Forest 1	1.72	3.75
Random Forest 2	1.72	3.74
XGBoost	2.56	3.81

As per above result summary of both Python & R, Random Forest 2 does well for R and, XGBoost perform best for Python.

❖ Conclusion

Feature engineering significantly improved the predictive ability of my Machine Learning model. Another way to improve the model's accuracy is to increase the amount of training data, and/or Hyper parameter tuning. If I have a lot of dimensions (features) in the data, dimensionality reduction techniques can also help improve the model's accuracy.

Thank You