New York City Taxi Fare Prediction

~SAPPARAPU RAHUL(IMT2016036)

PROBLEM STATEMENT

Task is predicting the fare amount (inclusive of tolls) for a taxi ride in New York City given the pickup and dropoff locations(with latitudes and longitudes).

DATA

ID: key - Unique string identifying each row in both the training and test sets. Comprised of pickup_datetime plus a unique integer, but this doesn't matter, it should just be used as a unique ID field. Required in your submission CSV. Not necessarily needed in the training set, but could be useful to simulate a 'submission file' while doing cross-validation within the training set.

Features

- **pickup_datetime - `timestamp` value indicating when the taxi ride started.**
- **pickup_longitude - `float` for longitude coordinate of where the taxi ride started.**
- **pickup_latitude - `float` for latitude coordinate of where the taxi ride started.**
- **dropoff_longitude - `float` for longitude coordinate of where the taxi ride ended.**
- **dropoff_latitude - `float` for latitude coordinate of where the taxi ride ended.**
- **passenger_count - an `integer` indicating the number of passengers in the taxi ride.**

Target

- **fare_amount - `float` dollar amount of the cost of the taxi ride. This value is only in the training set; this is what you are predicting in the test set and it is required in your submission CSV.**

EVALUATION METRIC

The evaluation metric for this competition is the root mean squared error or RMSE. RMSE measures the difference between the predictions of a model and the corresponding ground truth. A large RMSE is equivalent to a large average error, so smaller values of RMSE are better. One nice property of RMSE is that the error is given in the units being measured, so you can tell very directly how incorrect the model might be on unseen data.

RMSE is given by:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

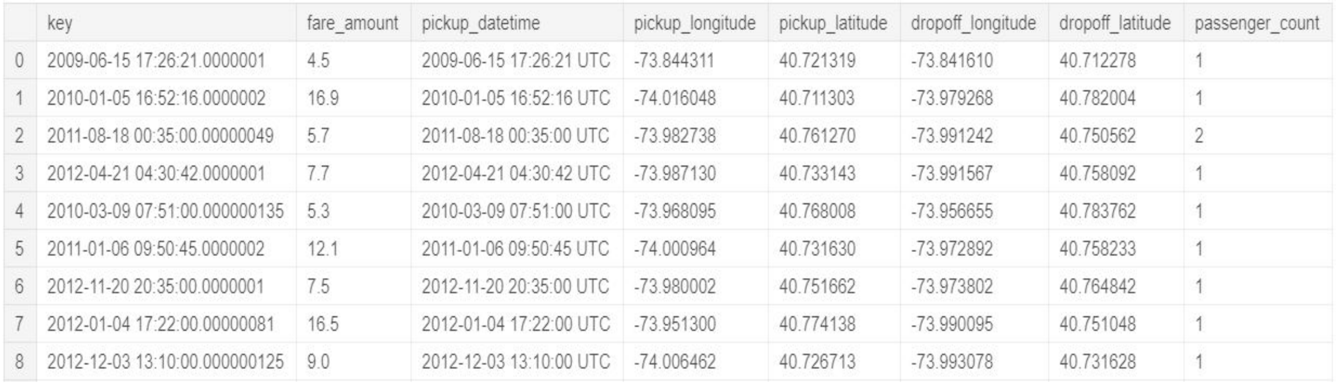
UNDERSTANDING DATA

Data is 55 million rows with 8 columns, out of which there are seven features and a unique id.

- The seven features are present to enable us to analyze and compute necessary extra fields from these data fields and create a robust model for prediction.
- The unique id is just to distinguish each tuple entry uniquely.

The task here is a regression problem as the job is to predict the estimated fare, which is a real number.

Glimpse of Data:



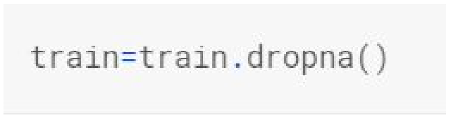
	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	2009-06-15 17:26:21.0000001	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1
1	2010-01-05 16:52:16.0000002	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1
2	2011-08-18 00:35:00.00000049	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2
3	2012-04-21 04:30:42.0000001	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1
4	2010-03-09 07:51:00.000000135	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1
5	2011-01-06 09:50:45.0000002	12.1	2011-01-06 09:50:45 UTC	-74.000964	40.731630	-73.972892	40.758233	1
6	2012-11-20 20:35:00.0000001	7.5	2012-11-20 20:35:00 UTC	-73.980002	40.751662	-73.973802	40.764842	1
7	2012-01-04 17:22:00.00000081	16.5	2012-01-04 17:22:00 UTC	-73.951300	40.774138	-73.990095	40.751048	1
8	2012-12-03 13:10:00.000000125	9.0	2012-12-03 13:10:00 UTC	-74.006462	40.726713	-73.993078	40.731628	1

DATA PREPROCESSING

In order for the data to consume, the data has to be cleaned and processed such that it is in a state when built model on such data, the model is expected to predict in robust and is as effective as possible.

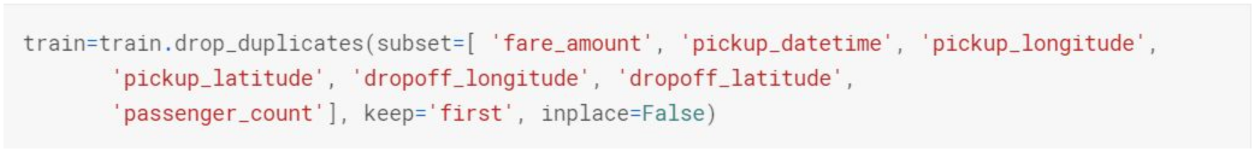
Primary Data cleaning:

- **Removing rows containing null values.**



```
train=train.dropna()
```

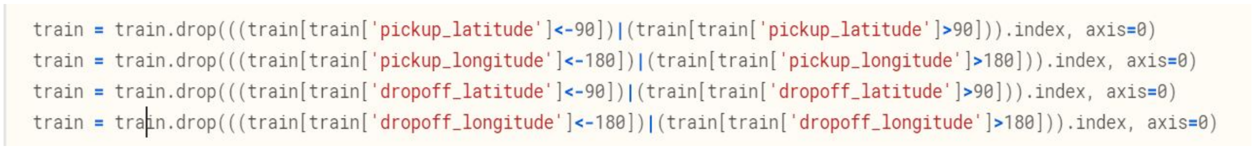
- **Removing duplicated tuples.**



```
train=train.drop_duplicates(subset=[ 'fare_amount', 'pickup_datetime', 'pickup_longitude',
        'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude',
        'passenger_count'], keep='first', inplace=False)
```

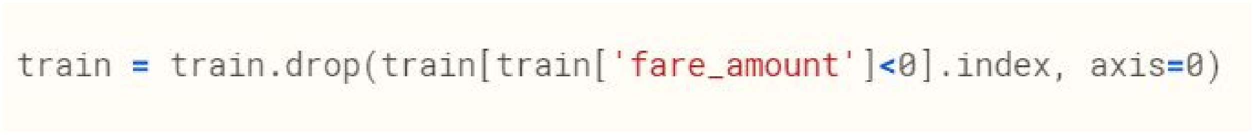
Secondary-level Data processing:

- **Clipping the range of latitudes and longitudes.**



```
train = train.drop(((train[train['pickup_latitude']<-90])|(train[train['pickup_latitude']>90])).index, axis=0)
train = train.drop(((train[train['pickup_longitude']<-180])|(train[train['pickup_longitude']>180])).index, axis=0)
train = train.drop(((train[train['dropoff_latitude']<-90])|(train[train['dropoff_latitude']>90])).index, axis=0)
train = train.drop(((train[train['dropoff_longitude']<-180])|(train[train['dropoff_longitude']>180])).index, axis=0)
```

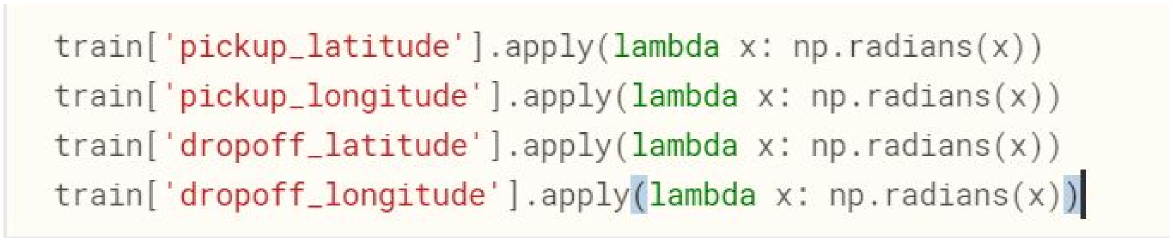
- **Removing the tuples who has invalid fare ranges.**



```
train = train.drop(train[train['fare_amount']<0].index, axis=0)
```

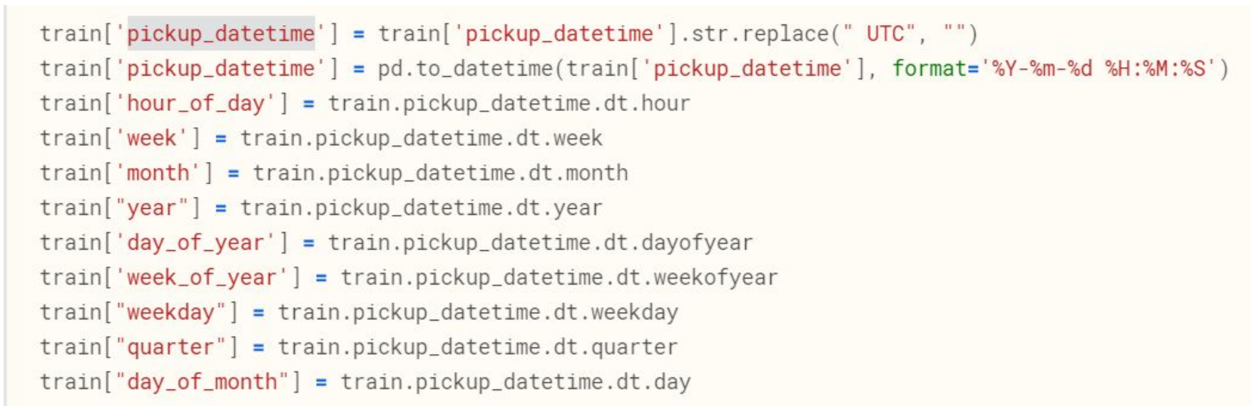
Tertiary Data Processing:

- **Normalizing the latitudes and longitudes data fields. (converting to radians)**



```
train['pickup_latitude'].apply(lambda x: np.radians(x))
train['pickup_longitude'].apply(lambda x: np.radians(x))
train['dropoff_latitude'].apply(lambda x: np.radians(x))
train['dropoff_longitude'].apply(lambda x: np.radians(x))
```

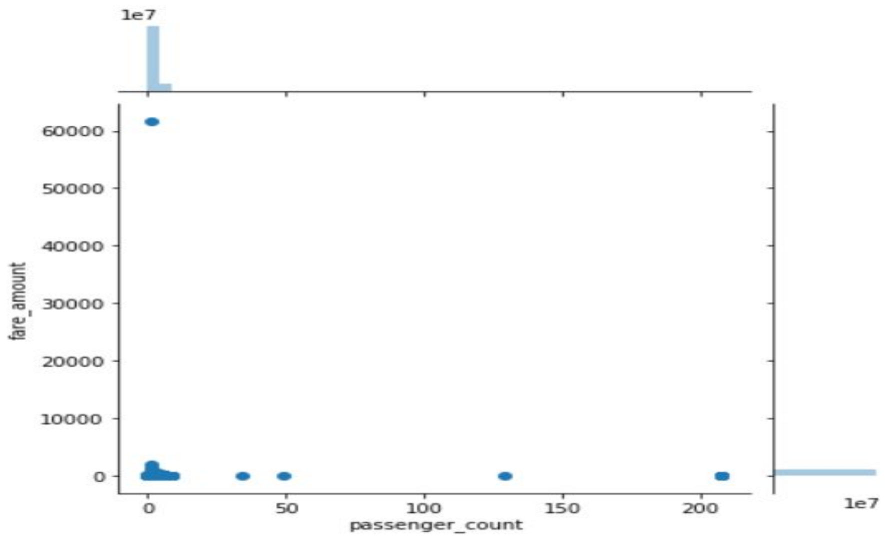
- **Splitting the Data field 'pickup_datetime' into different time factors(day, month, year etc)**



```
train['pickup_datetime'] = train['pickup_datetime'].str.replace(" UTC", "")
train['pickup_datetime'] = pd.to_datetime(train['pickup_datetime'], format='%Y-%m-%d %H:%M:%S')
train['hour_of_day'] = train.pickup_datetime.dt.hour
train['week'] = train.pickup_datetime.dt.week
train['month'] = train.pickup_datetime.dt.month
train["year"] = train.pickup_datetime.dt.year
train['day_of_year'] = train.pickup_datetime.dt.dayofyear
train['week_of_year'] = train.pickup_datetime.dt.weekofyear
train["weekday"] = train.pickup_datetime.dt.weekday
train["quarter"] = train.pickup_datetime.dt.quarter
train["day_of_month"] = train.pickup_datetime.dt.day
```

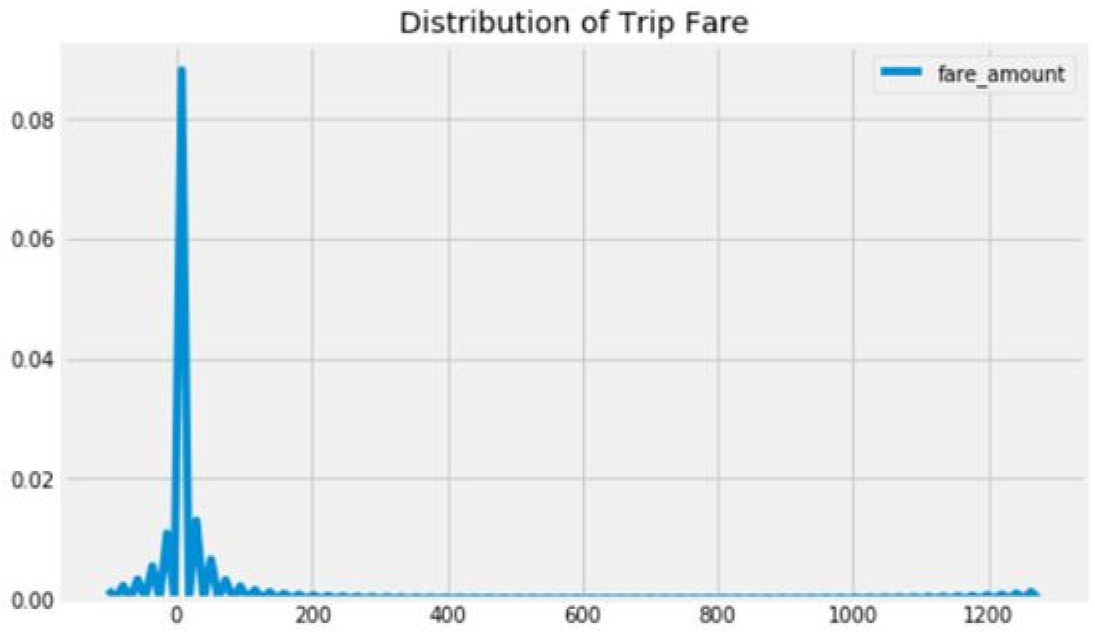
Removing Outliers:

Clipping fare_amount values in the range frequency of [0 - 10000]



Explorative Data Analysis(EDA)

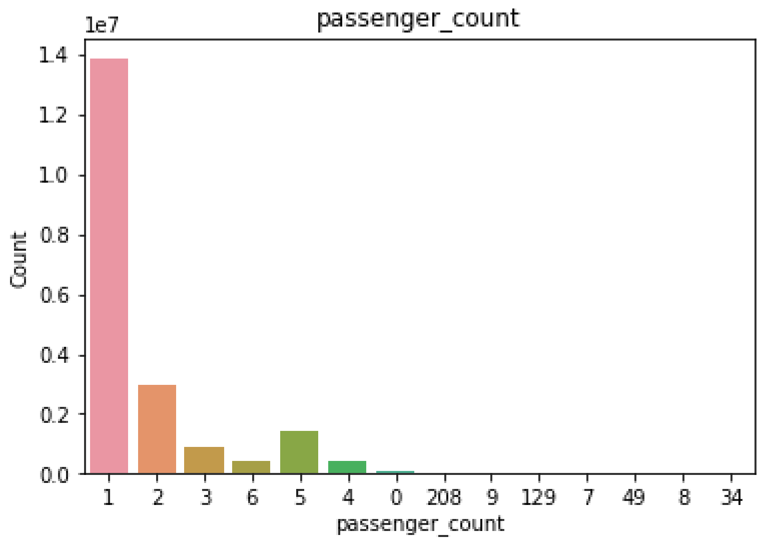
- **Observation: Distribution of Trip Fare**
- **Analysis**: As pictorially represented, the fare is distributed over a wide range but concentrated in the range [10-50].
- It implies that the frequency of the trips of minimal fare is very high(implication of that peak in the plot)



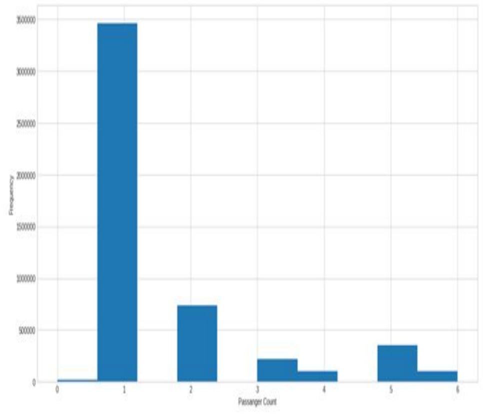
- **Observation: Passenger Count Frequency**

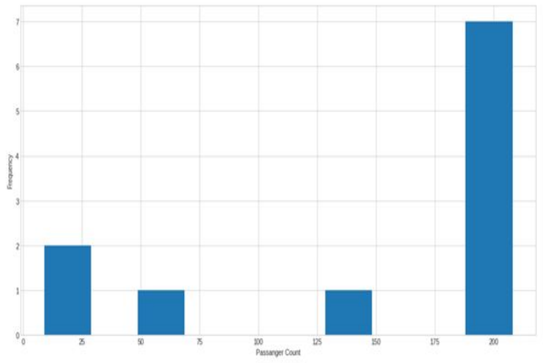
  Analysis: As plotted, the number of passengers travelling in a single taxi is in the range [0-5], but out of which single passenger frequency is very minimal.Also, passenger_count greater than 7 is almost negligible.concentrated in the range [10-50].

- It implies that the frequency of the passengers generally varies from one to five,with one passenger count having maximum weightage as seen from plots below.



-

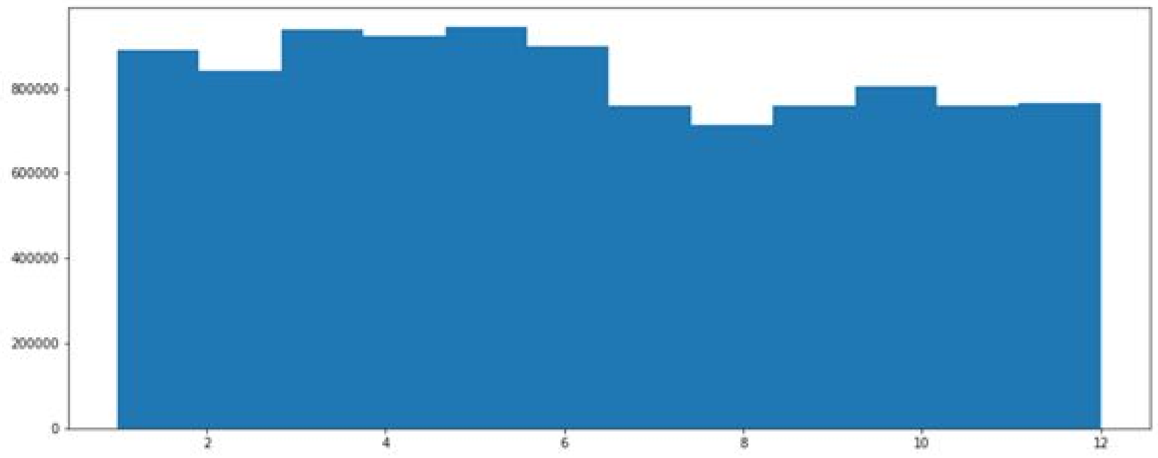




Passenger_count<7 *Passenger_count>7*

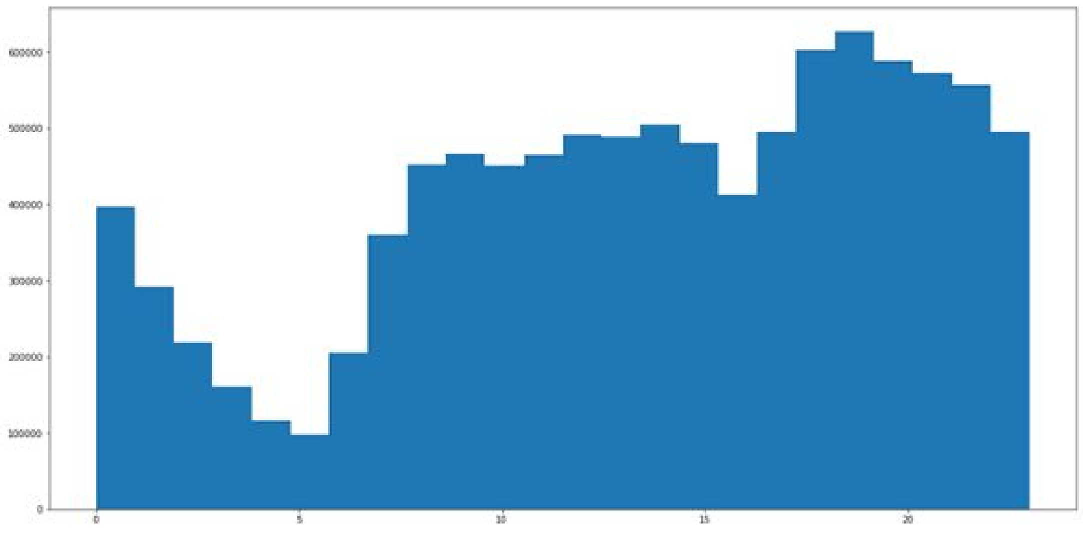
- **Observation: frequency of months occurrence in trips**

Analysis: As seen, the frequency of trips is more or less the same magnitude and decently high every month. It states that the variation of difference in its contribution is very minimal, hence it cannot be used as a factor of prediction(feature)

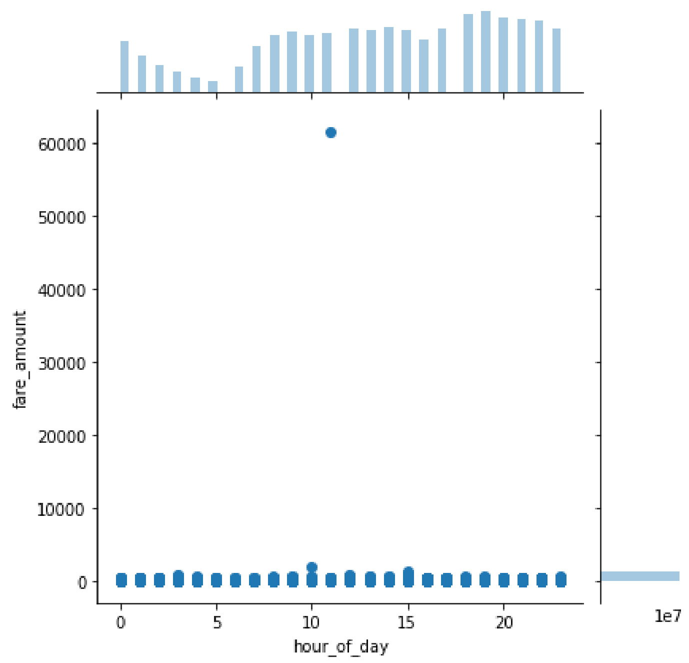


- **Observation: Frequency of days occurrence in trips**

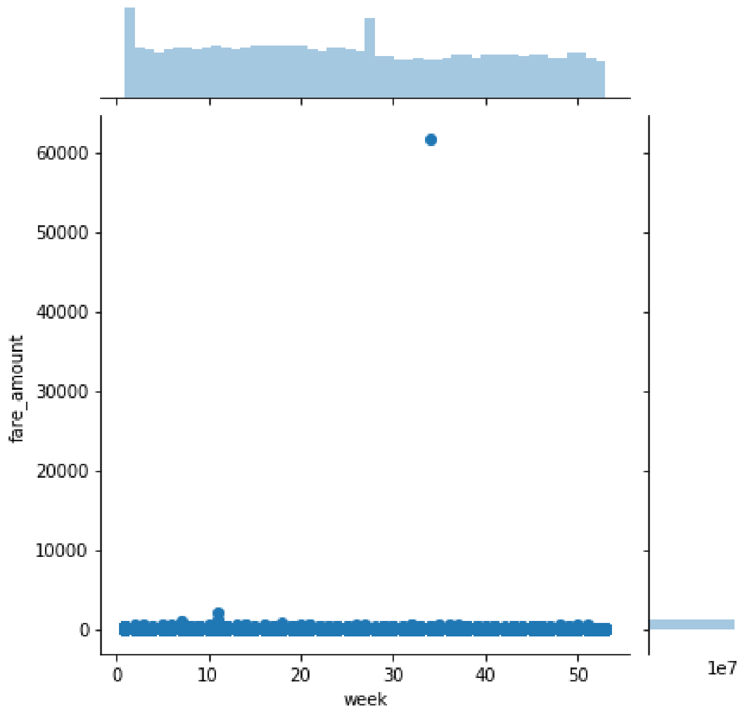
Analysis: There is a very slight variance of occurrence of trips in the days of the month, hence it could possibly a feature to consider.



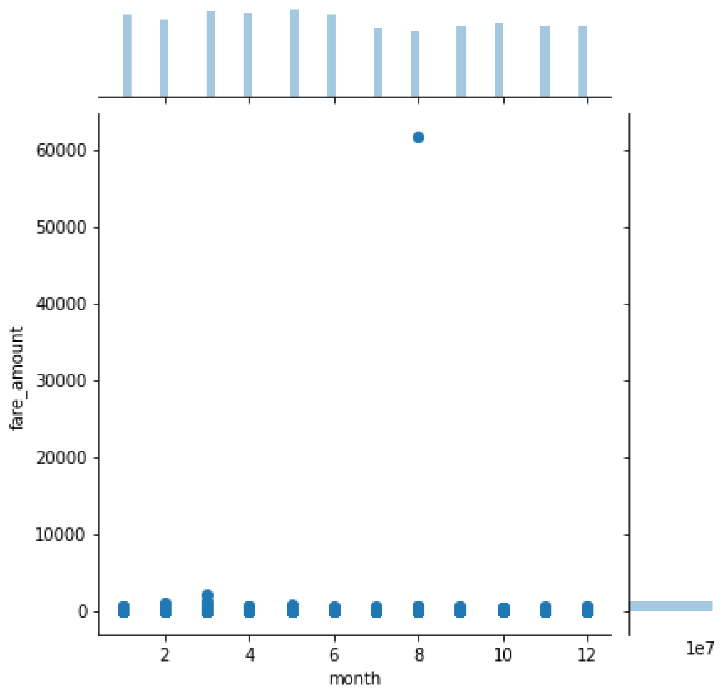
Co-Relation between DataFields:



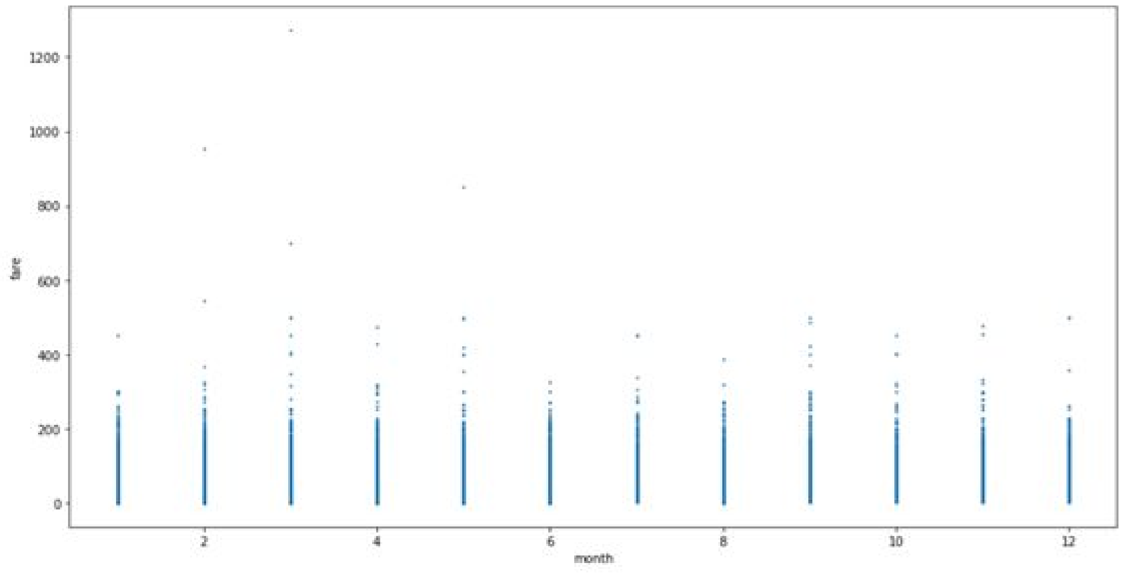
Fare v/s Hour of day

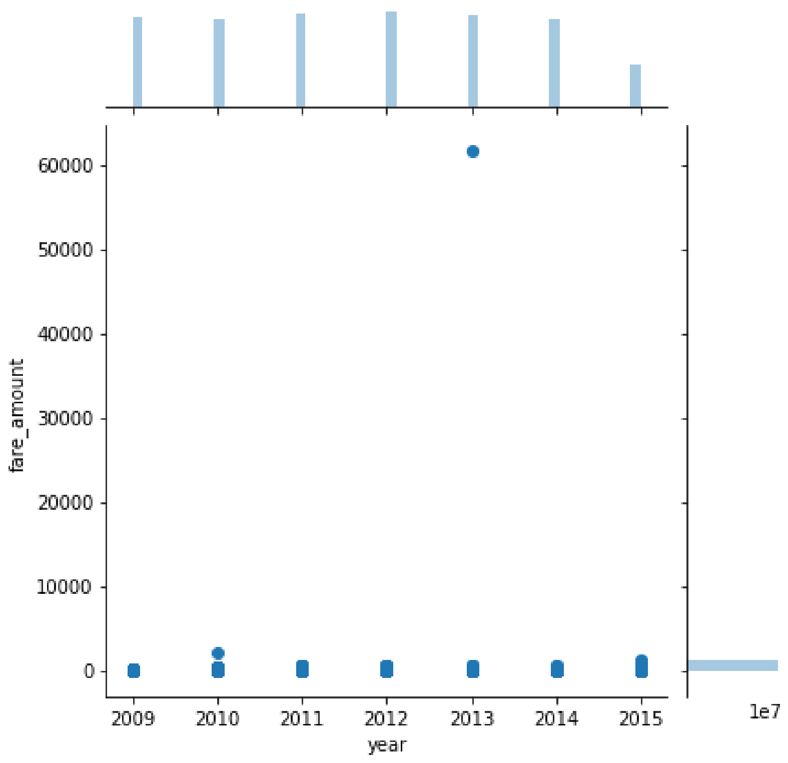


Fare v/s Week



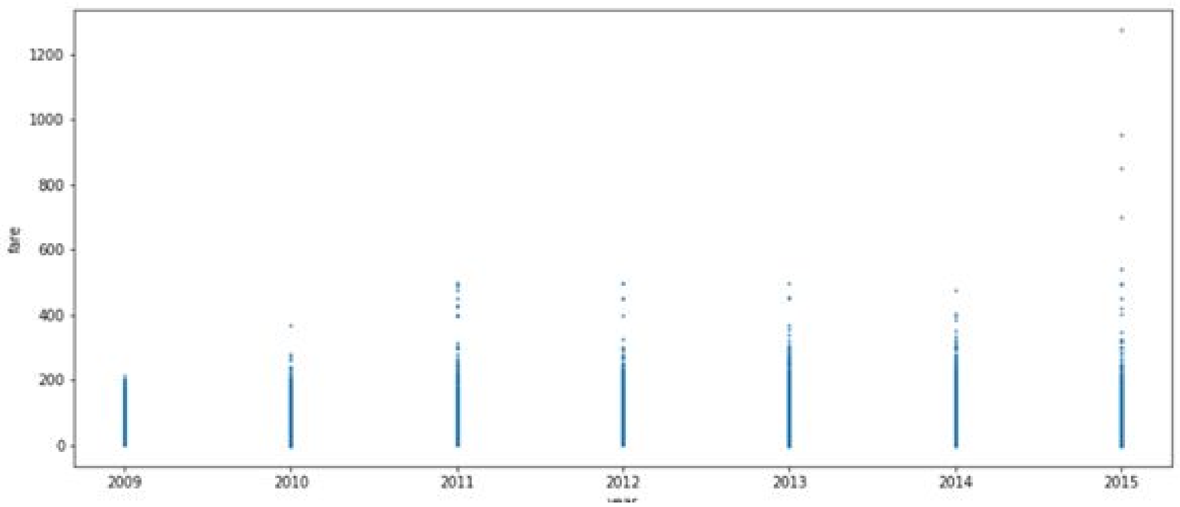
Fare v/s Month



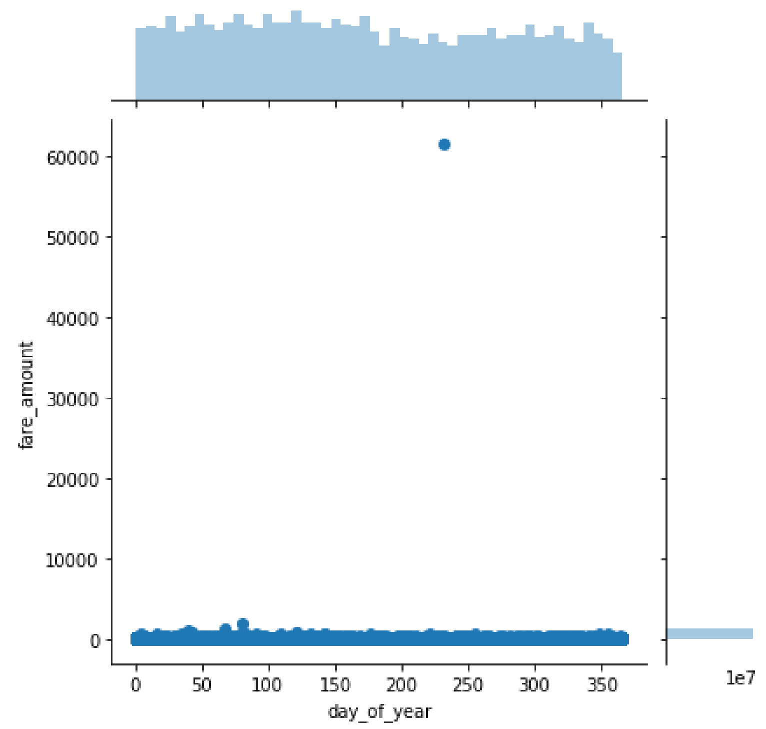


60000
50000
40000
30000
20000
10000
0
fare_amount
2009
2010
2011
2012
2013
2014
2015
year
1e7

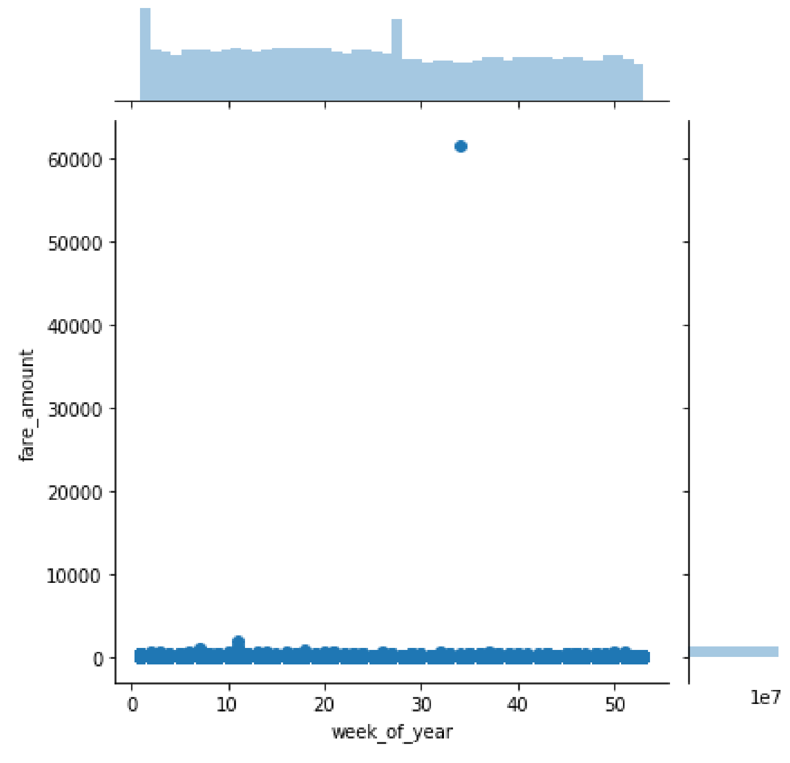
Fare v/s Year



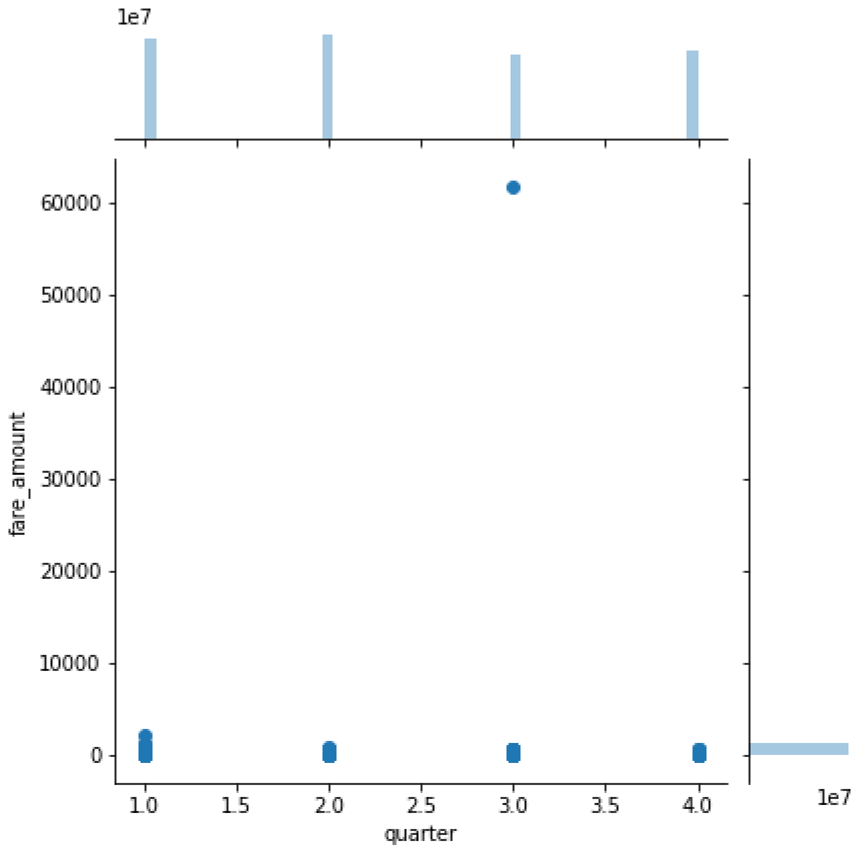
1200
1000
800
600
400
200
0
fare
2009
2010
2011
2012
2013
2014
2015
year



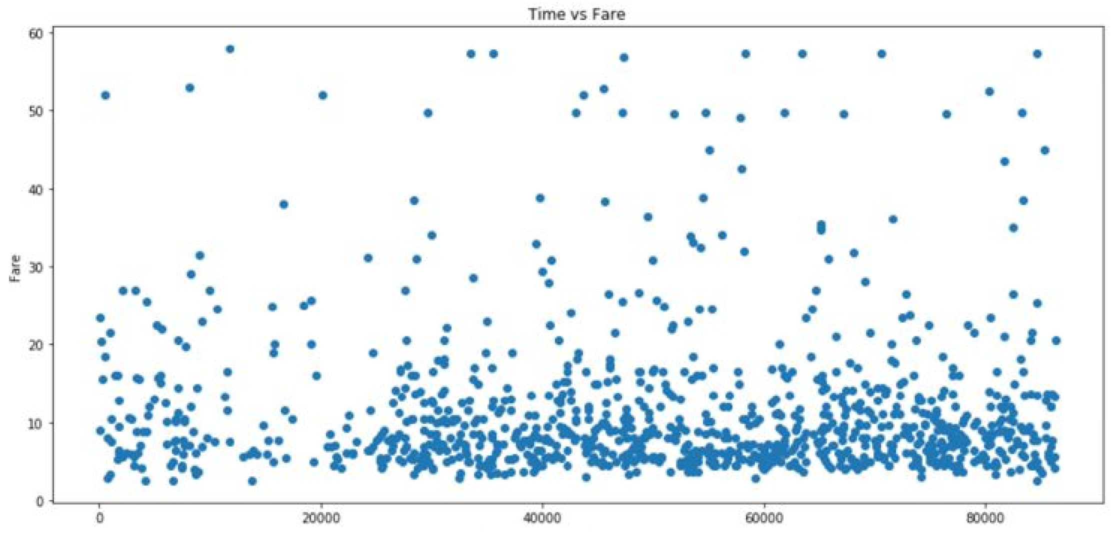
Fare v/s Day of Year



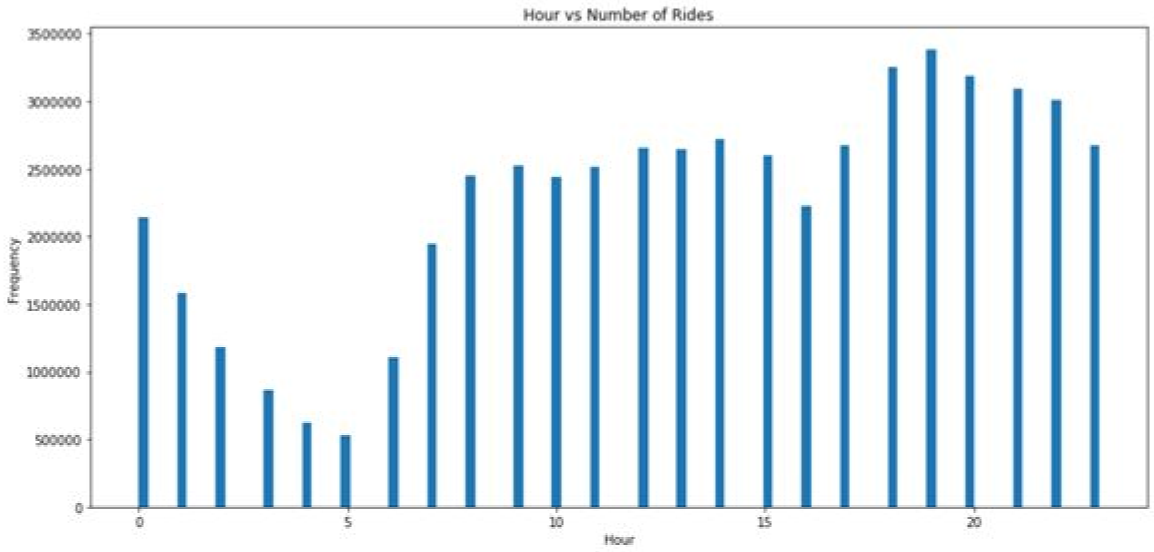
Fare v/s Week of Year

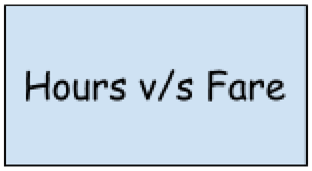


Fare v/s Quarter of Year



Time v/s Fare





Observations:

- Every data field has a decent contribution to the variation of Fare amount, but some dominate whereas some don't.

- In the plots considering month and quarter, there seems that these both don't stand as proper choice of the feature as there is not a much significant variation of the target field.

- Rest all remain step forward to get selected as features for model building and can sustain as factors of prediction.

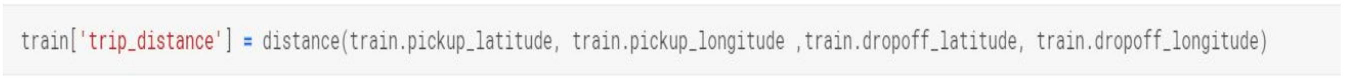
- Hence Exploratory data analysis provides a sense of judgement to extract the possible features that can help us build the required model satisfying mentioned specifications.

FEATURE EXTRACTION

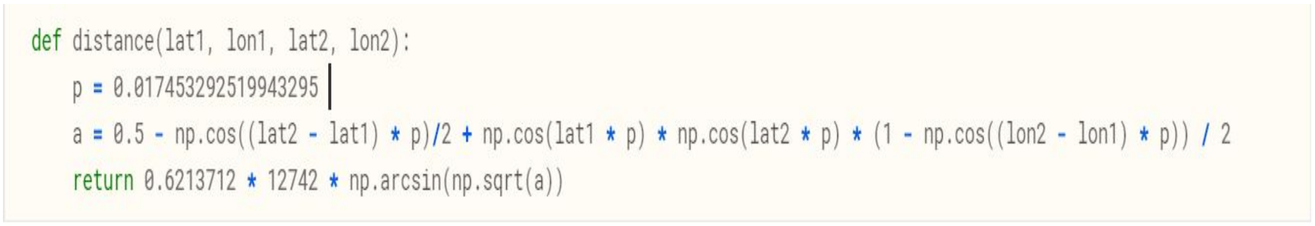
Initially, the features used for model building are decided just by my understanding of data and proceeded with a sense of my instincts, but soon realized this is not the case that has to be featured. But with the performance of models and analysis of results understood that there are some unnecessary features that have to be removed and some effective features should be added.

With that understanding and the analysis of data the primary feature that developed is <u>trip_distance</u>.

- **Trip distance is the measure to find the distance between the pickup and dropoff locations using latitudes and longitudes considering also the direction of displacement. The measure is called Haversine Distance.**



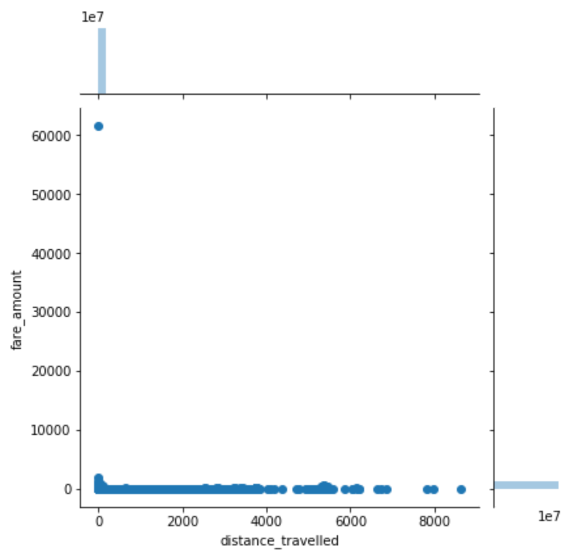
```
train['trip_distance'] = distance(train.pickup_latitude, train.pickup_longitude ,train.dropoff_latitude, train.dropoff_longitude)
```



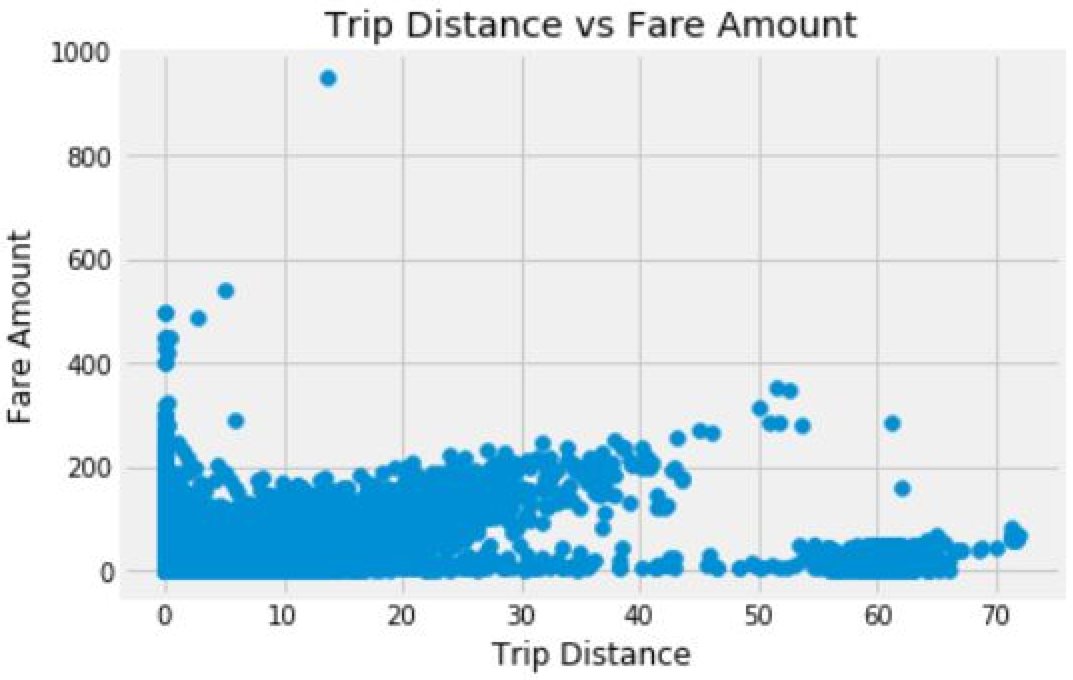
```
def distance(lat1, lon1, lat2, lon2):
    p = 0.017453292519943295
    a = 0.5 - np.cos((lat2 - lat1) * p)/2 + np.cos(lat1 * p) * np.cos(lat2 * p) * (1 - np.cos((lon2 - lon1) * p)) / 2
    return 0.6213712 * 12742 * np.arcsin(np.sqrt(a))
```

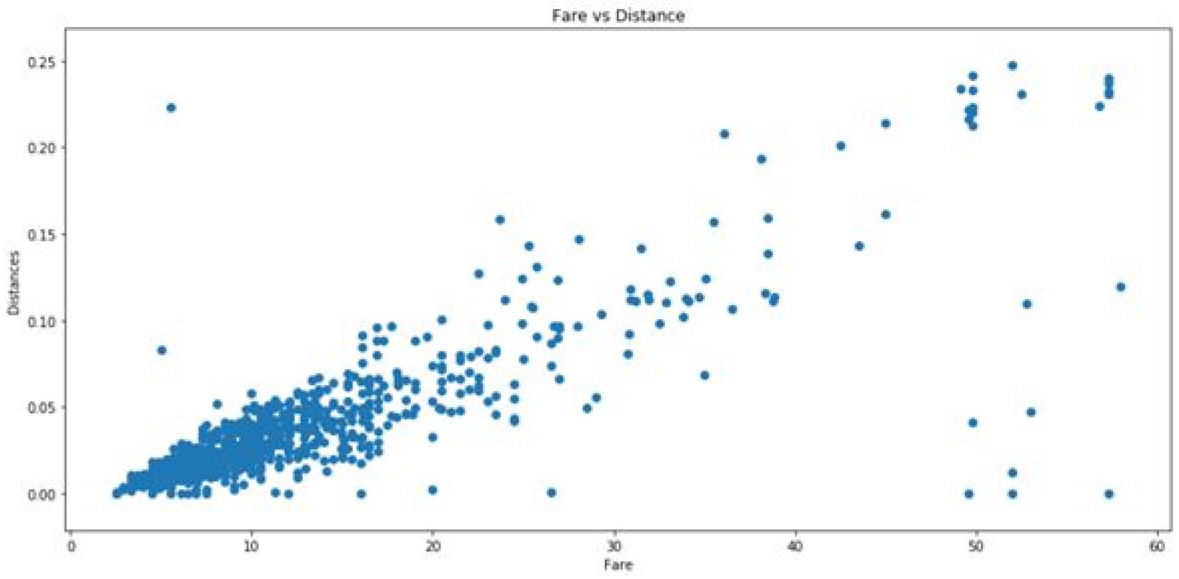
<u>**Evidence to show that selection of this feature is appropriate:**</u>

- Dependence plot between fare and trip distance travelled:



- **Scatter plot between the target(fare) and trip distance (in both orientations)**





Observations:

- Trip distance is directly co-related with fare distance and there is significant variation o fare with trip distance.Hence considered as an option for selecting as a primary feature.

Features Extracted:

As per Exploratory Data Analysis(EDA), some features are extracted as primary features, some as secondary and some are rejected as per the conclusions are drawn from the analysis and behaviour of data.

Primary features:

- Passenger_count
- Trip_distance
- pickup _latitude/longitude
- dropoff_latitude/longitude
- Hour

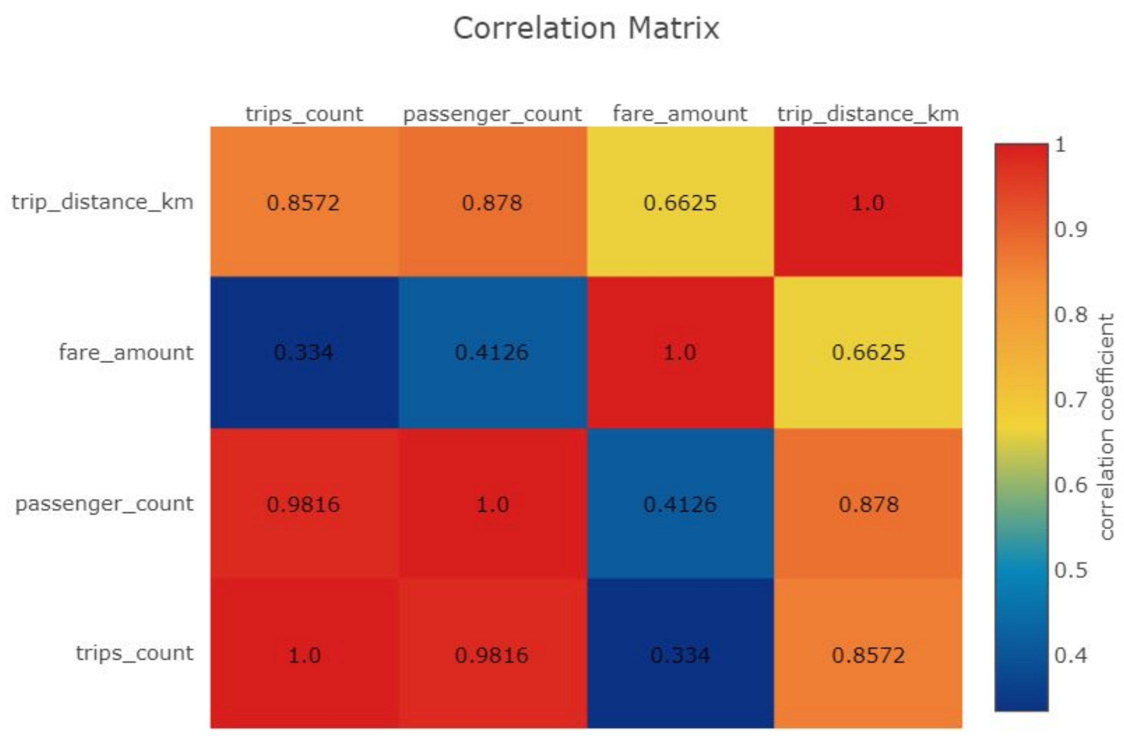
Secondary features:

- Day
- weekday

Discarded Features:

- Week_of_year
- Quarter
- Month
- year

Co-Relation Matrix between primary Features:



MODEL SELECTION

Due to the heavy variation and diverse behaviour of data and bulky magnitude of data, the data seems to demand tree models for robust productivity and effectiveness.

I have tried three classical models. Out of which gradient boosting models worked better.

- Random Forest
- Boosting Trees:
  - ➢ Xg-Boost
  - ➢ Lightgbm

- *Random Forest regressor:*

```
from sklearn.ensemble  import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=120, max_features=0.5, min_samples_leaf=3, n_jobs=-1, oob_score=False)
rf.fit(train, y)
y_test_pred=rf.predict(test)
print(y_test_pred)
```

- *LightGBM:*

```
import lightgbm as lgb
lgb = lgb.LGBMRegressor(objective='regression',num_leaves=35, n_estimators=300)

lgb.fit(train, y)
y_test_pred=lgb.predict(test)
print(y_test_pred)
```

- *XgBoost:*

```
import xgboost
xg =xgboost.XGBRegressor(n_estimators=300,max_depth=15,max_leaves=9,random_state=0)
xg.fit(train,y)
y_test_pred=xg.predict(test)
print(y_test_pred)
```

<u>Best Outputs from Each model:</u>

RandomForest:

```
test-rmse:4.0133
```

LightGBM:

```
test-rmse:3.67111
```

Xgboost:

```
test-rmse:3.63908
```

CONCLUSION

Therefore, after analyzing from the analysis of data from Explorative data analysis and examining the behaviour of data to extract the appropriate features to be factors of prediction and using them to build a model on top of that satisfying mentioned requirements and specifications, the **gradient boosting model(Xgboost)** and considered robust model as per the process carried above with the accuracy of: **3.63908.**

By:-

SAPPARAPU RAHUL

IMT2016036