PROJECT REPORT
ON
Cab Fare Prediction

Deepak Mathur

Contents

Introduction

Problem Statement :

The objective of this project is to predict Cab Fare amount. 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.

Data :

Attributes: ·

• pickup_datetime - timestamp value indicating when the cab ride started.

• pickup_longitude - float for longitude coordinate of where the cab ride started.

• pickup_latitude - float for latitude coordinate of where the cab ride started.

• dropoff_longitude - float for longitude coordinate of where the cab ride ended.

• dropoff_latitude - float for latitude coordinate of where the cab ride ended.

• passenger_count - an integer indicating the number of passengers in the cab ride.

Methodology

Pre-Processing :

Data pre-processing is the first stage of any type of project. In this stage we get the feel of the data. We do this by looking at plots of independent variables vs target variables. If the data is messy, we try to improve it by sorting deleting extra rows and columns. This stage is called as Exploratory Data Analysis. This stage generally involves data cleaning, merging, sorting, looking for outlier analysis, looking for missing values in the data, imputing missing values if found by various methods such as mean, median, mode, KNN imputation, etc.

Removing values which are not within desired range(outlier) depending upon basic understanding of dataset :

If everything is dropped which are beyond range then below are the missing percentages for each variable:

	Variables	Missing_percentage
0	passenger_count	0.351191
1	fare_amount	0.140476
2	pickup_datetime	0.006385
3	pickup_longitude	0.000000
4	pickup_latitude	0.000000
5	dropoff_longitude	0.000000
6	dropoff_latitude	0.000000

After imputing above mentioned missing values kNN algorithm values at a particular row which was made nan using np.nan method

```
fare_amount           7.3698
pickup_longitude    -73.9954
pickup_latitude      40.7597
dropoff_longitude   -73.9876
dropoff_latitude     40.7512
passenger_count            2
```

Missing value Analysis:

In this step we look for missing values in the dataset like empty row column cell which was left after removing special characters and punctuation marks. Some missing values are in form of NA. missing values left behind after outlier analysis; missing values can be in any form. Unfortunately, in this dataset we have found some missing values. Therefore, we will do some missing value analysis. Before imputed we selected random row no-1000 and made it NA, so that we will compare original value with imputed value and choose best method which will impute value closer to actual value.

	index	0
0	fare_amount	22
1	pickup_datetime	1
2	pickup_longitude	0
3	pickup_latitude	0
4	dropoff_longitude	0
5	dropoff_latitude	0
6	passenger_count	55

We will impute values for fare_amount and passenger_count both of them has missing values 22 and 55 respectively. We will drop 1 value in pickup_datetime i.e it will be an entire row to drop.
Below are the missing value percentage for each variable:

	Variables	Missing_percentage
0	passenger_count	0.351191
1	fare_amount	0.140476
2	pickup_datetime	0.006385
3	pickup_longitude	0.000000
4	pickup_latitude	0.000000
5	dropoff_longitude	0.000000
6	dropoff_latitude	0.000000

And below is the Standard deviation of particular variable which has missing values in them:

```
fare_amount          435.982171
passenger_count        1.266096
```

We’d tried central statistical methods and algorithmic method--KNN to impute missing values in the dataset:

For Passenger_count:

Actual value = 1

Mode = 1

KNN = 2

We will choose the KNN method here because it maintains the standard deviation of variable. We will not use Mode method because whole variable will be more biased towards 1 passenger_count also passenger_count has maximum value equals to 1

.For Passenger_count:

Actual value = 7.0

Mean = 15.118

Median = 8.5

KNN = 7.369801

We will Choose KNN method here because it imputes value closest to actual value also it maintains the Standard deviation of the variable.

Standard deviation for passenger_count and fare_amount after KNN imputation:

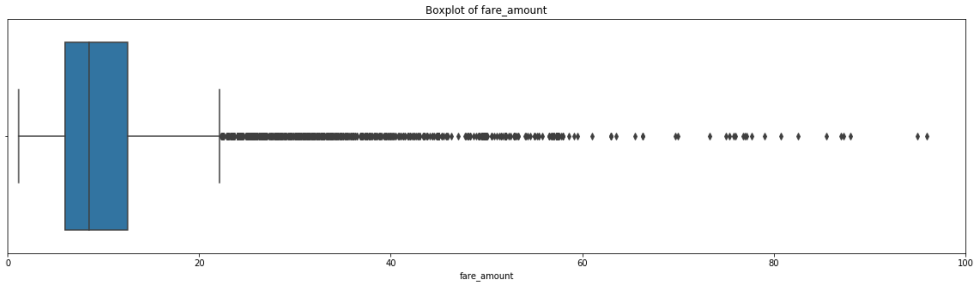
```
fare_amount          435.661995
passenger_count        1.264322
dtype: float64
```

Outlier Analysis

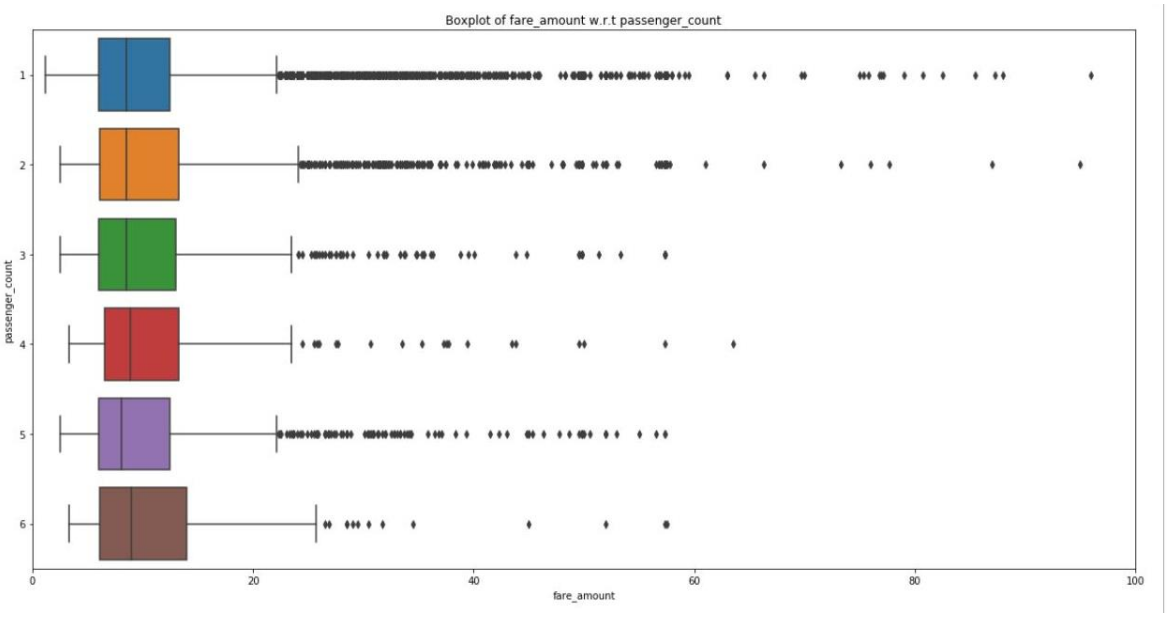
We look for outlier in the dataset by plotting Boxplots. There are outliers present in the data. we have removed these outliers. This is how we done, I. We replaced them with Nan values or we can say created missing values. II. Then we imputed those missing values with KNN method.

• We Will do Outlier Analysis only on Fare_amount just for now and we will do outlier analysis after feature engineering latitudes and longitudes.

• Univariate Boxplots: Boxplots for target variable. Univariate Boxplots: Boxplots for all Numerical Variables also for target variable



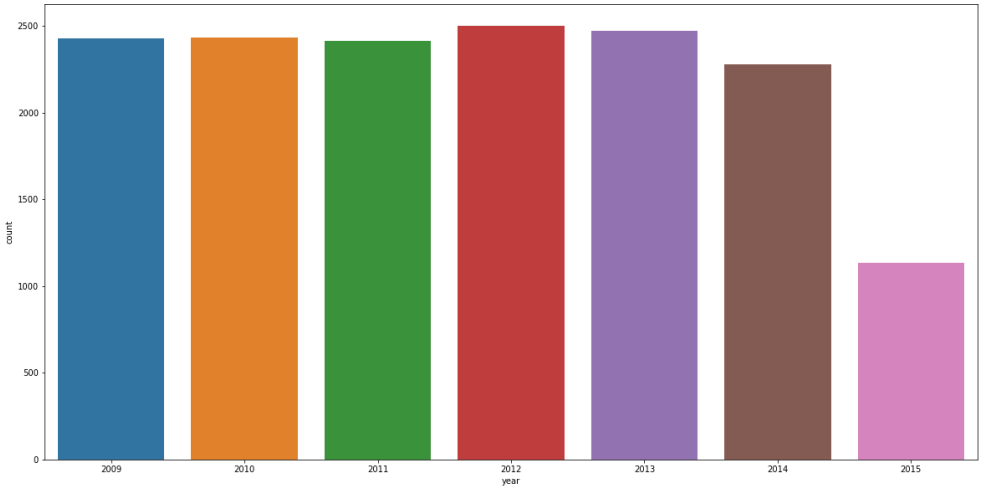
Bivariate Boxplots: Boxplots for all fare_amount Variables Vs all passenger_count variable.

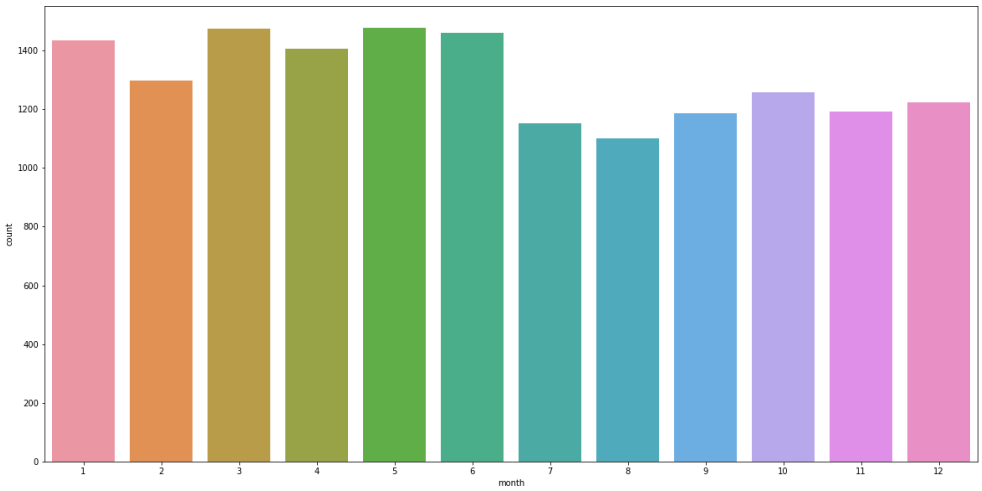


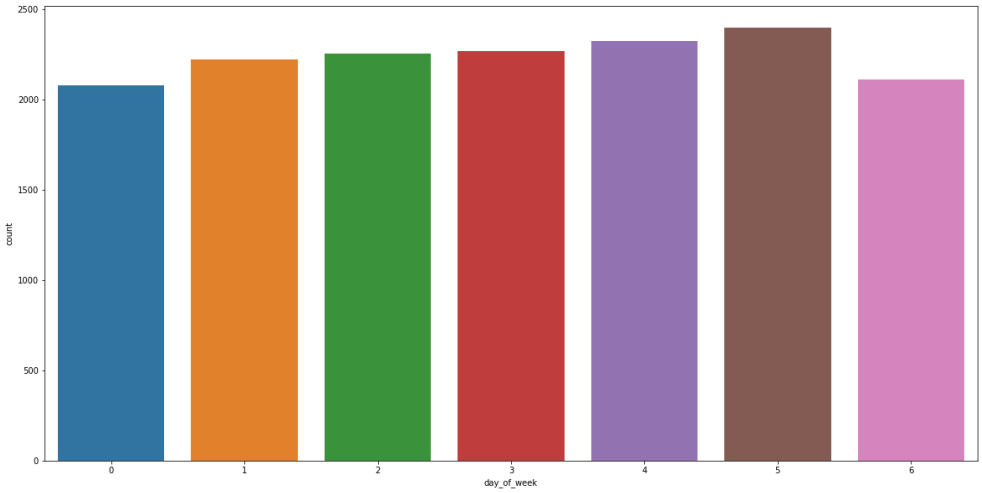
From above Boxplots we see that ‘fare_amount’have outliers in it: ‘fare_amount’ has 1359 outliers. We successfully imputed these outliers with KNN and K value is 3

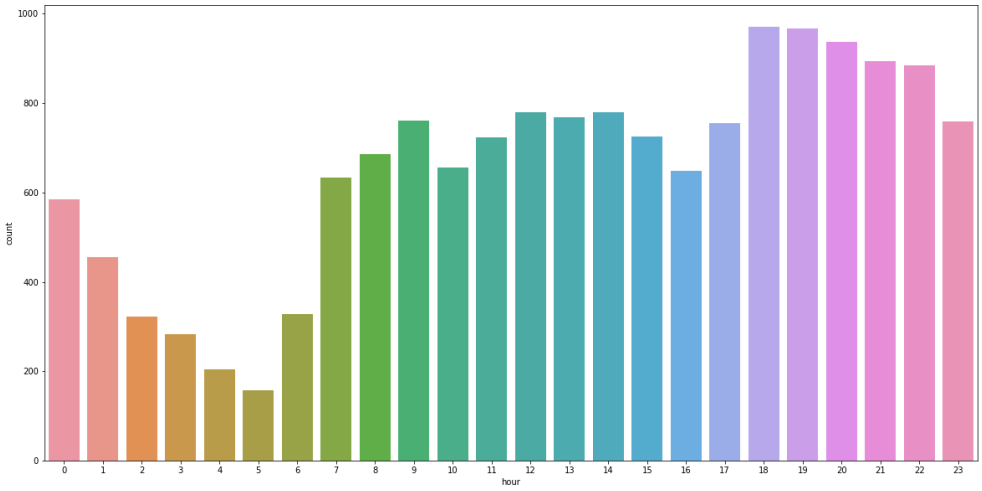
Feature Engineering

Feature Engineering is used to drive new features from existing features. 1. For ‘pickup_datetime’ variable: We will use this timestamp variable to create new variables. New features will be year, month, day_of_week, hour. ‘year’ will contain only years from pickup_datetime. For ex. 2009, 2010, 2011, etc. ‘month’ will contain only months from pickup_datetime. For ex. 1 for January, 2 for February, etc. ‘day_of_week’ will contain only week from pickup_datetime. For ex. 1 which is for Monday,2 for Tuesday,etc. ‘hour’ will contain only hours from pickup_datetime. For ex. 1, 2, 3, etc.









As we have now these new variables we will categorize them to new variables like Session from hour column, seasons from month column, week:weekday/weekend from day_of_week variable. So, session variable which will contain categories—morning, afternoon, evening, night_PM, night_AM. Seasons variable will contain categories—spring, summer, fall, winter. Week will contain categories—weekday, weekend.

For ‘Latitudes’ and ‘Longitudes’ variables:

As we have latitude and longitude data for pickup and dropoff, we will find the distance the cab travelled from pickup and dropoff location and for vincenty, new variable name will be ‘geodesic’.

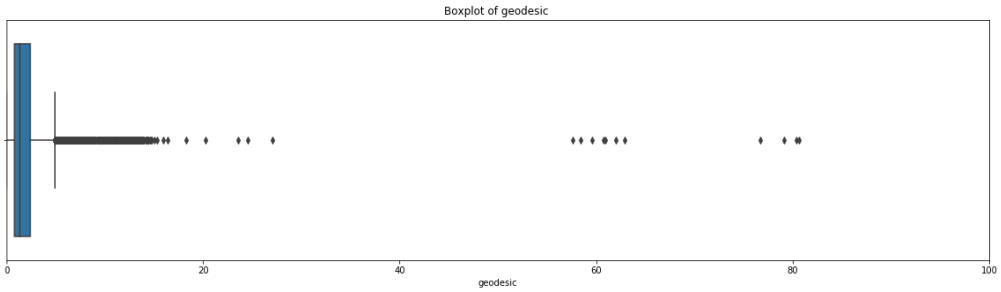
As Vincenty is more accurate than haversine. Also, vincenty is prefered for short distances. Therefore, we will drop great_circle.

```
'fare_amount', 'passenger_count_1', 'passenger_count_2',
'passenger_count_3', 'passenger_count_4', 'passenger_count_5',
'passenger_count_6', 'season_fall', 'season_spring', 'season_summer',
'season_winter', 'week_weekday', 'week_weekend', 'session_afternoon',
'session_evening', 'session_morning', 'session_night_AM',
'session_night_PM', 'year_2009', 'year_2010', 'year_2011', 'year_2012',
'year_2013', 'year_2014', 'year_2015', 'geodesic'
```

Columns in testing data after feature engineering:

```
'passenger_count_1', 'passenger_count_2', 'passenger_count_3',
'passenger_count_4', 'passenger_count_5', 'passenger_count_6',
'season_fall', 'season_spring', 'season_summer', 'season_winter',
'week_weekday', 'week_weekend', 'session_afternoon', 'session_evening',
'session_morning', 'session_night_AM', 'session_night_PM', 'year_2009',
'year_2010', 'year_2011', 'year_2012', 'year_2013', 'year_2014',
'year_2015', 'geodesic'
```

we will plot boxplot for our new variable ‘geodesic’: Boxplot of ‘geodesic’ for range 0 to 100 miles.



Feature Selection

In this step we would allow only to pass relevant features to further steps. We remove irrelevant features from the dataset. We do this by some statistical techniques, like we look for features which will not be helpful in predicting the target variables. In this dataset we have to predict the fare_amount.

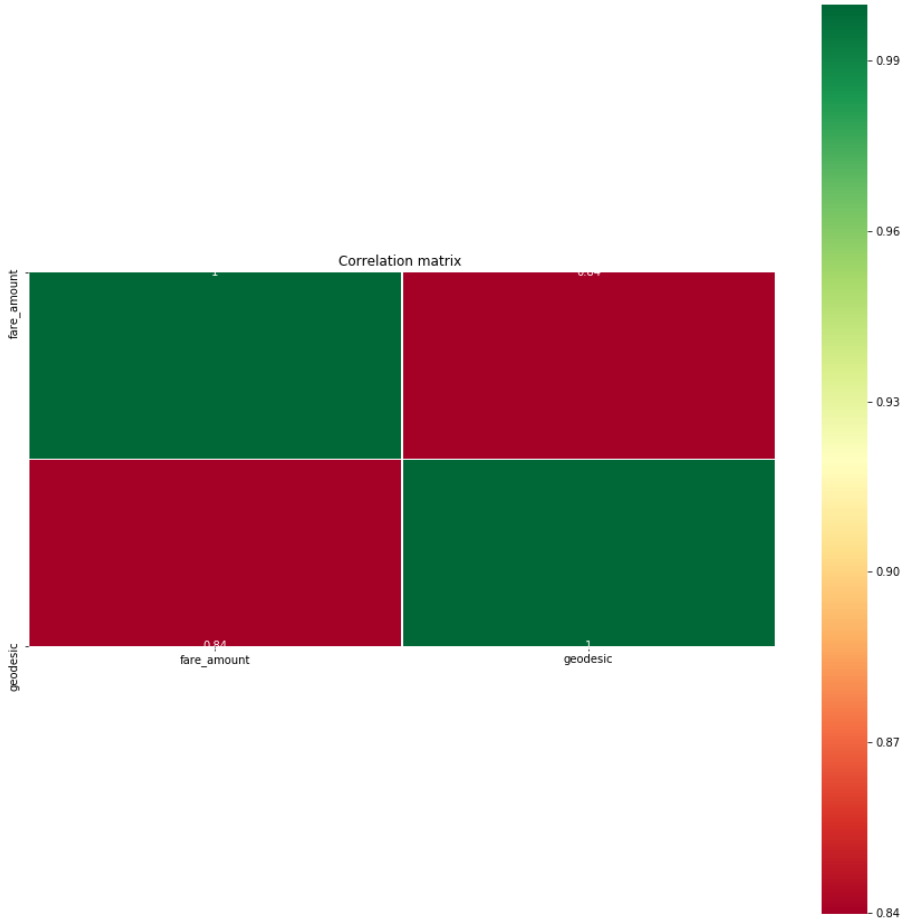
Further below are some types of test involved for feature selection:

1 Correlation analysis – This requires only numerical variables. Therefore, we will filter out only numerical variables and feed it to correlation analysis. We do this by plotting correlation plot for all numerical variables. There should be no correlation between independent variables but there should be high correlation between independent variable and dependent variable. So, we plot the correlation plot. we can see that in correlation plot faded colour like skin colour indicates that 2 variables are highly correlated with each other. As the colour fades correlation values increases. From below correlation plot we see that:

• 'fare_amount' and 'geodesic' are very highly correlated with each other.

• As fare_amount is the target variable and ‘geodesic’ is independent variable we will keep ‘geodesic’ because it will help to explain variation in fare_amount.

Correlation Plot:



Chi-Square test of independence – Unlike correlation analysis we will filter out only categorical variables and pass it to Chi-Square test. Chi-square test compares 2 categorical variables in a contingency table to see if they are related or not.

I. Assumption for chi-square test: Dependency between Independent variable and dependent variable should be high and there should be no dependency among independent variables.

II. II. Before proceeding to calculate chi-square statistic, we do the hypothesis testing: Null hypothesis: 2 variables are independent. Alternate hypothesis: 2 variables are not independent.

The interpretation of chi-square test:

I. For theorical or excel sheet purpose: If chi-square statistics is greater than critical value then reject the null hypothesis saying that 2 variables are dependent and if it's less, then accept the null hypothesis saying that 2 variables are independent.

II. While programming: If p-value is less than 0.05 then we reject the null hypothesis saying that 2 variables are dependent and if p-value is greater than 0.05 then we accept the null hypothesis saying that 2 variables are independent. Here we did the test between categorical independent variables pairwise

Analysis of Variance(Anova) Test –

- It is carried out to compare between each group in a categorical variable.
- ANOVA only lets us know the means for different groups are same or not. It doesn't help us identify which mean is different.

Hypothesis testing: -

Null Hypothesis: mean of all categories in a variable are same.

Alternate Hypothesis: mean of at least one category in a variable is different.

If p-value is less than 0.05 then we reject the null hypothesis.

And if p-value is greater than 0.05 then we accept the null hypothesis.

Below is the anova analysis table for each categorical variable :

	df	sum_sq	mean_sq	F	PR(>F)
C(passenger_count_2)	1.0	10.568728	10.568728	0.545736	4.600765e-01
C(passenger_count_3)	1.0	17.059092	17.059092	0.880877	3.479760e-01
C(passenger_count_4)	1.0	63.933288	63.933288	3.301312	6.924368e-02
C(passenger_count_5)	1.0	21.286924	21.286924	1.099189	2.944603e-01
C(passenger_count_6)	1.0	145.816804	145.816804	7.529518	6.076534e-03
C(season_spring)	1.0	28.945378	28.945378	1.494648	2.215152e-01
C(season_summer)	1.0	26.849975	26.849975	1.386448	2.390238e-01
C(season_winter)	1.0	481.677256	481.677256	24.872287	6.191324e-07
C(week_weekend)	1.0	130.826572	130.826572	6.755470	9.354849e-03
C(session_night_AM)	1.0	2130.002751	2130.002751	109.986594	1.200118e-25
C(session_night_PM)	1.0	185.393568	185.393568	9.573137	1.977933e-03
C(session_evening)	1.0	0.984476	0.984476	0.050835	8.216192e-01
C(session_morning)	1.0	48.799442	48.799442	2.519849	1.124416e-01
C(year_2010)	1.0	1507.605557	1507.605557	77.847975	1.228675e-18
C(year_2011)	1.0	1332.016859	1332.016859	68.781131	1.188955e-16
C(year_2012)	1.0	431.078196	431.078196	22.259512	2.402360e-06
C(year_2013)	1.0	340.775242	340.775242	17.596554	2.745881e-05
C(year_2014)	1.0	1497.202720	1497.202720	77.310805	1.610520e-18
C(year_2015)	1.0	2587.658975	2587.658975	133.618511	8.830922e-31
Residual	15640.0	302884.577261	19.366022	NaN	NaN

Looking at above table every variable has p value less than 0.05 so reject the null hypothesis.

Multicollinearity :

In regression, "multicollinearity" refers to predictors that are correlated with other predictors. Multicollinearity occurs when your model includes multiple factors that are correlated not just to your response variable, but also to each other.

I. Multicollinearity increases the standard errors of the coefficients.
II. Increased standard errors in turn means that coefficients for some independent variables may be found not to be significantly different from 0.
III. In other words, by overinflating the standard errors, multicollinearity makes some variables statistically insignificant when they should be significant. Without multicollinearity (and thus, with lower standard errors), those coefficients might be significant.
IV. VIF is always greater or equal to 1. if VIF is 1 --- Not correlated to any of the variables. if VIF is between 1-5 --- Moderately correlated. if VIF is above 5 --- Highly correlated. If there are multiple variables with VIF greater than 5, only remove the variable with the highest VIF.
V. And if the VIF goes above 10, you can assume that the regression coefficients are poorly estimated due to multicollinearity.

Below is the table for VIF analysis for each independent variable:

	VIF	features
0	15.269760	Intercept
1	1.040699	passenger_count_2[T.1.0]
2	1.019514	passenger_count_3[T.1.0]
3	1.011860	passenger_count_4[T.1.0]
4	1.024990	passenger_count_5[T.1.0]
5	1.017215	passenger_count_6[T.1.0]
6	1.642257	season_spring[T.1.0]
7	1.552391	season_summer[T.1.0]
8	1.587603	season_winter[T.1.0]
9	1.050861	week_weekend[T.1.0]
10	1.376870	session_night_AM[T.1.0]
11	1.423418	session_night_PM[T.1.0]
12	1.524842	session_evening[T.1.0]
13	1.559218	session_morning[T.1.0]
14	1.691349	year_2010[T.1.0]
15	1.687697	year_2011[T.1.0]
16	1.711042	year_2012[T.1.0]
17	1.709335	year_2013[T.1.0]
18	1.664978	year_2014[T.1.0]
19	1.406919	year_2015[T.1.0]
20	1.025980	geodesic

We have checked for multicollinearity in our Dataset and all VIF values are below 5.

Feature Scaling

Data Scaling methods are used when we want our variables in data to scaled on common ground. It is performed only on continuous variables.

• Normalization: Normalization refer to the dividing of a vector by its length. normalization normalizes the data in the range of 0 to 1. It is generally used when we are planning to use distance method for our model development purpose such as KNN. Normalizing the data improves convergence of such algorithms. Normalisation of data scales the data to a very small interval, where outliers can be loosed.

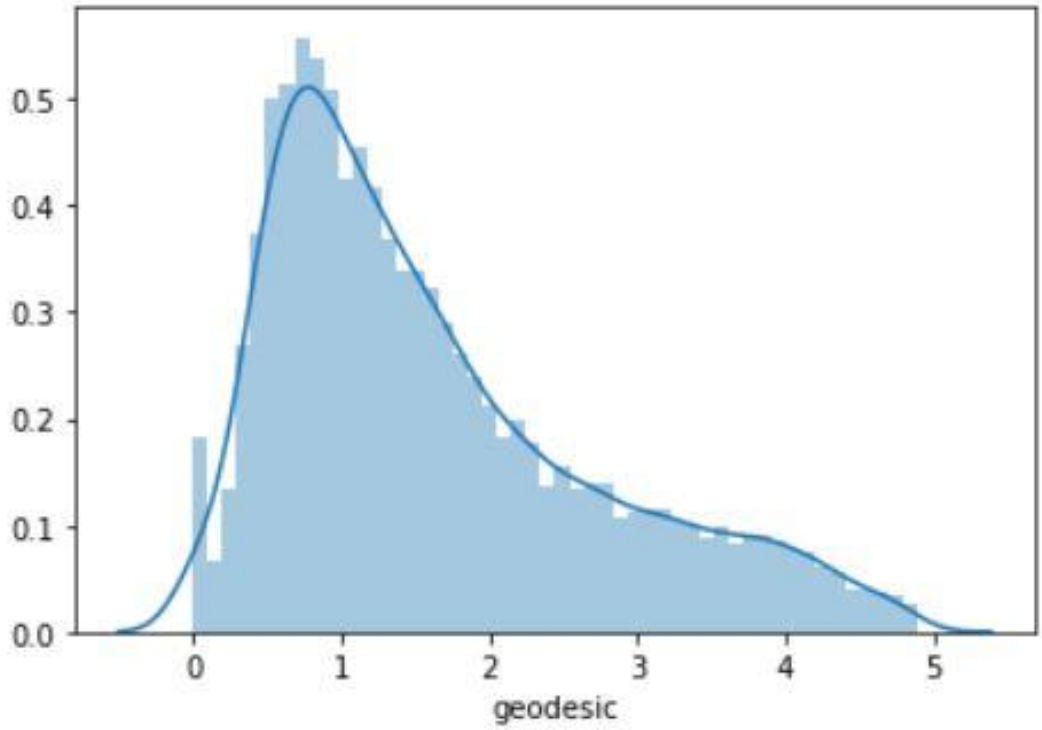
• Standardization: Standardization refers to the subtraction of mean from individual point and then dividing by its SD. Z is negative when the raw score is below the mean and Z is positive when above mean. When the data is distributed normally you should go for standardization.

Linear Models assume that the data you are feeding are related in a linear fashion, or can be measured with a linear distance metric. Also, our independent numerical variable ‘geodesic’ is not distributed normally so we had chosen normalization over standardization.

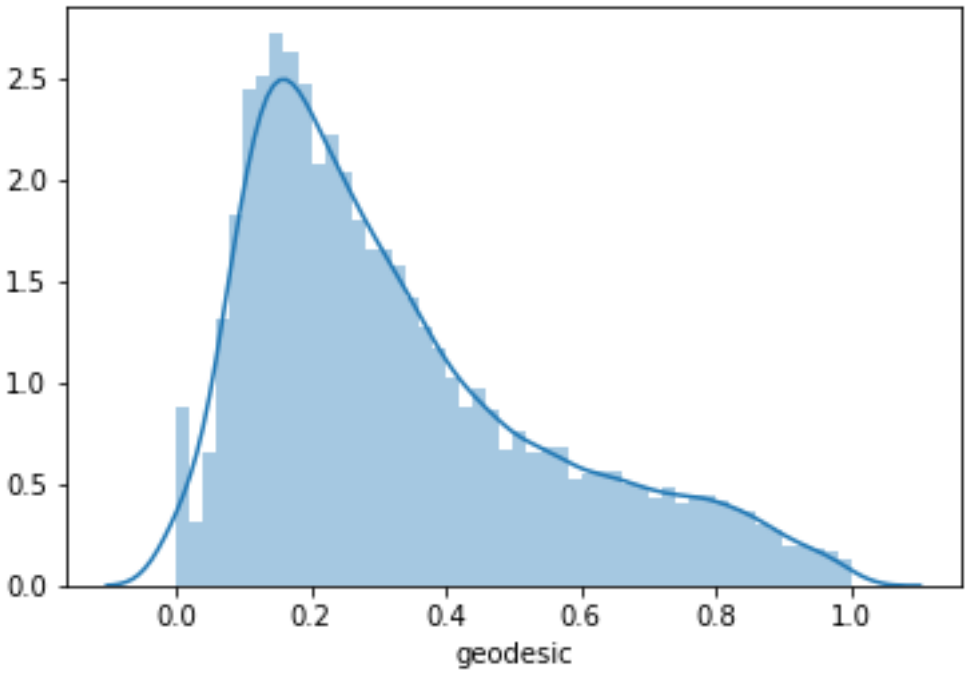
We have checked variance for each column in dataset before Normalisation.

High variance will affect the accuracy of the model. So, we want to normalise that variance. Graphs based on which standardization was chosen:

Note: It is performed only on Continuous variables. distplot() for ‘geodesic’ feature before normalization



distplot() for ‘geodesic’ feature after normalization:



Splitting train and Validation Dataset

a) We have used sklearn’s train_test_split() method to divide whole Dataset into train and validation datset.

b) 25% is in validation dataset and 75% is in training data.

c) 11745 observations in training and 3915 observations in validation dataset.

d) We will test the performance of model on validation datset.

e) The model which performs best will be chosen to perform on test dataset provided along with original train dataset.

f) X_train y_train--are train subset.

g) X_test y_test--are validation subset

Hyperparameter Optimization

a. To find the optimal hyperparameter we have used sklearn.model_selection.GridSearchCV. and sklearn.model_selection.RandomizedSearchCV

b. GridSearchCV tries all the parameters that we provide it and then returns the best suited parameter for data.

c. We gave parameter dictionary to GridSearchCV which contains keys which are parameter names and values are the values of parameters which we want to try for.

Below are best hyperparameter we found for different models:

Ridge Regression:

```
Tuned Decision ridge Parameters: {'alpha': 0.0004498432668969444, 'max_
iter': 500, 'normalize': True}
Best score is 0.732485216428362
```

Lasso Regression:

```
Tuned Decision lasso Parameters: {'alpha': 0.0011513953993264468,
'max_iter': 500, 'normalize': False}

Best score is 0.732519633822799
```

Decision Tree Regression:

```
Tuned Decision Tree Parameters: {'max_depth': 6, 'min_samples_split': 2
}
Best score is 0.7287411728769153
```

Random Forest Regression:

```
Tuned Random Forest Parameters: {'n_estimators': 200, 'min_samples_spli
t': 3, 'min_samples_leaf': 3, 'max_features': 'auto', 'max_depth': 13,
'bootstrap': True}
Best score is 0.7423876283274332
```

Xgboost regression:

```
Tuned Xgboost Parameters: {'subsample': 0.1, 'reg_alpha': 0.08685113737
513521, 'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.05, 'co
lsample_bytree': 0.7000000000000001, 'colsample_bynode': 0.700000000000
0001, 'colsample_bylevel': 0.9000000000000001}
Best score is 0.7476294983965251
```

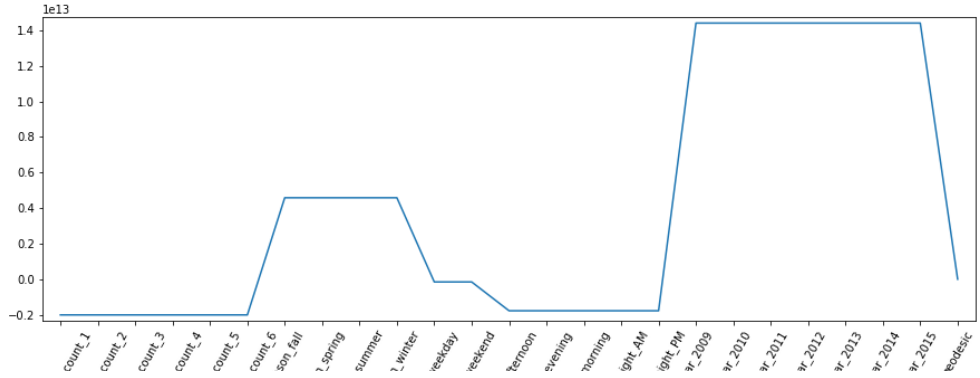
Model Development

Our problem statement wants us to predict the fare_amount. This is a Regression problem. So, we are going to build regression models on training data and predict it on test data. In this project I have built models using 5 Regression Algorithms:

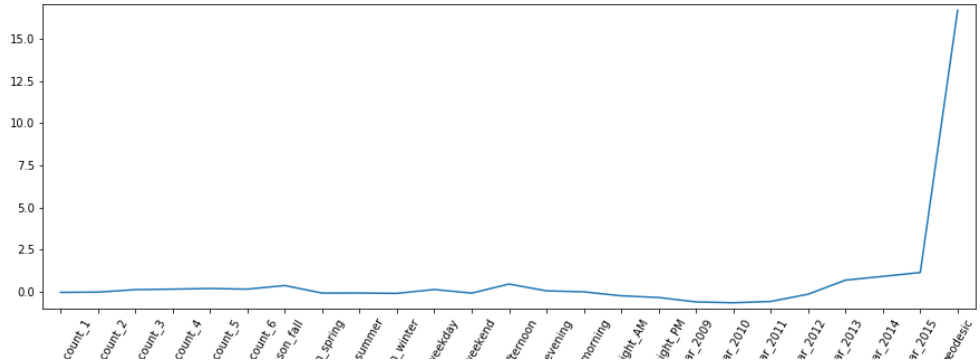
I. Linear Regression
II. Ridge Regression
III. Lasso Regression
IV. Decision Tree
V. Random Forest
VI. Xgboost Regression

Model Performance :

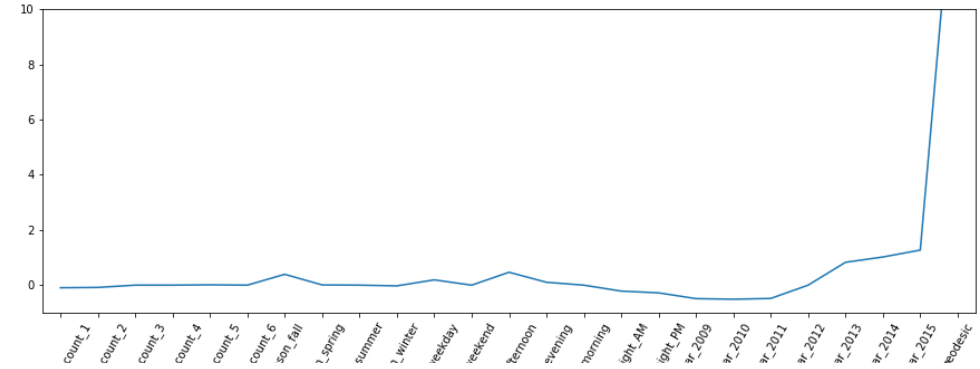
Multiple Linear Regression :



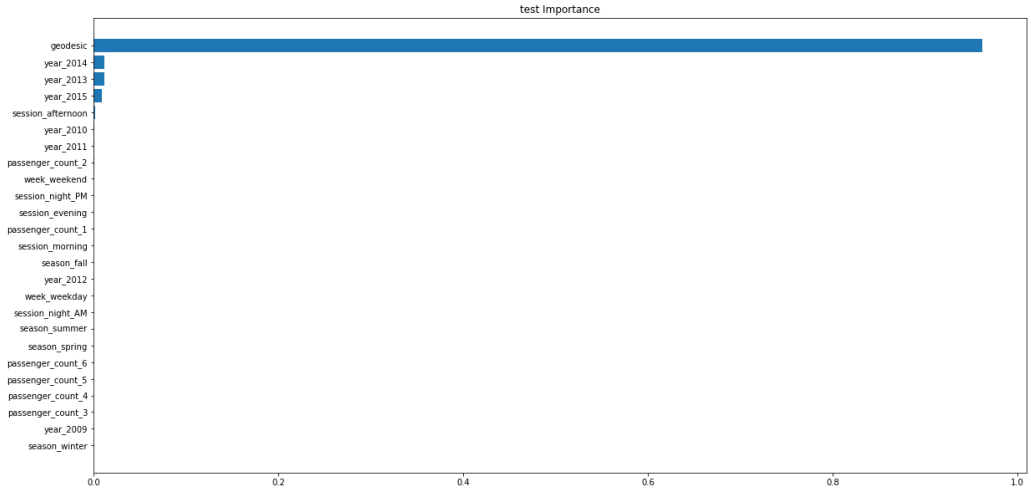
Ridge Regression:



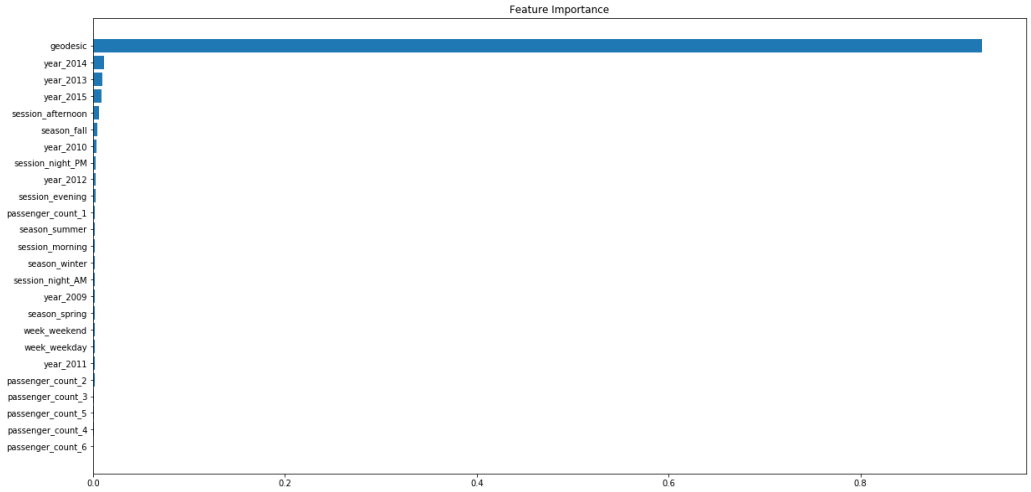
Lasso Regression:



Decision Tree Regression:



Random Forest Regression:




```
[-5.1113495  -5.09286622 -5.04831669 -5.11356383 -5.15293999]
Average 5-Fold CV Score: -5.103807246546199
```

Improving accuracy

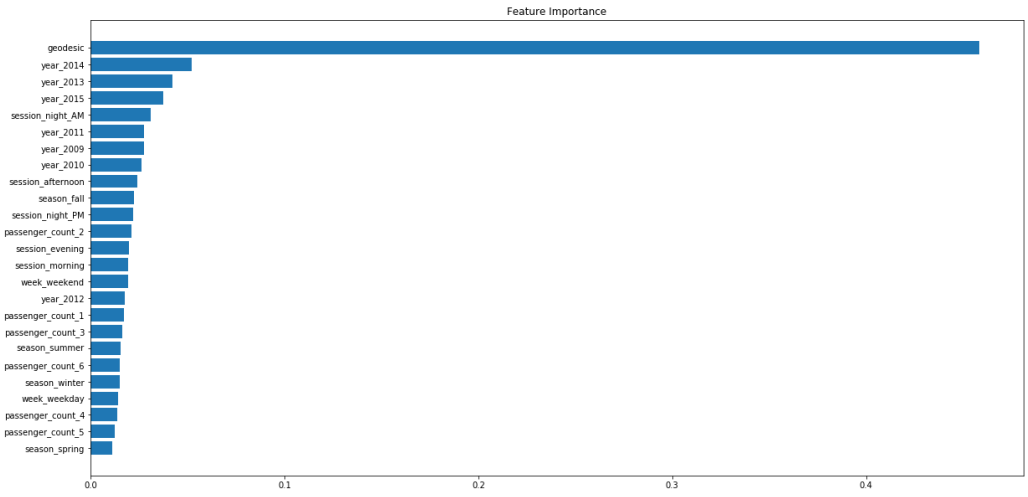
Improve Accuracy

a) Algorithm Tuning

b) Ensembles

We have used xgboost as a ensemble technique.

```
Tuned Xgboost Parameters: {'subsample': 0.1, 'reg_alpha': 0.08685113737
513521, 'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.05, 'co
lsample_bytree': 0.7000000000000001, 'colsample_bynode': 0.700000000000
0001, 'colsample_bylevel': 0.9000000000000001}
```



Finalize model

We have trained a Xgboost model on entire training dataset and used that model to predict on test data.Also, we have saved model for later use.

```
<<<-------------------- Training Data Score ---------------------->

r square   0.7542601370011739
Adjusted r square:0.7537359031437654
MAPE:18.030639321949952
MSE: 4.887567502409207
RMSE: 2.2107843636160465
RMSLE: 0.20422186998519856
```