CAB FARE PREDICTION

- Rahul kumar

Contents

Chapter 1

Introduction

1.1 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.

1.2 Problem Description

Number of 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.

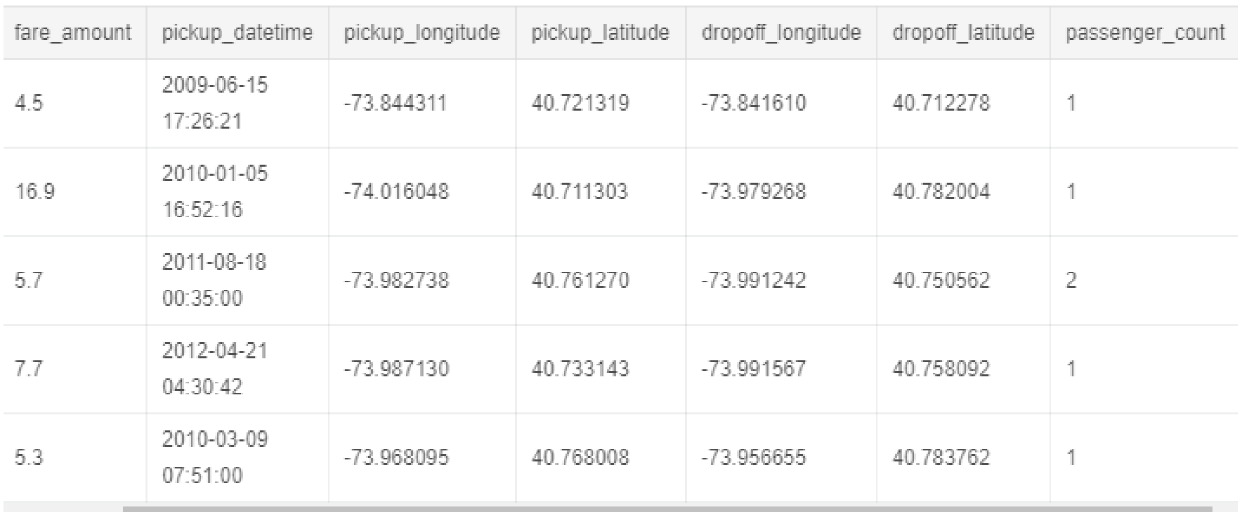
dropoff_longitude - float for longitude coordinate of where the cab ride ended.

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.

1.3 Data

The data is a Time-Series data but instead we will approach it as Regression Problem.Our task is to build a regression model which will predict the fare of the cab facility based on the customer attributes and all of the information during the journey and general information available to the company about them.



fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
4.5	2009-06-15 17:26:21	-73.844311	40.721319	-73.841610	40.712278	1
16.9	2010-01-05 16:52:16	-74.016048	40.711303	-73.979268	40.782004	1
5.7	2011-08-18 00:35:00	-73.982738	40.761270	-73.991242	40.750562	2
7.7	2012-04-21 04:30:42	-73.987130	40.733143	-73.991567	40.758092	1
5.3	2010-03-09 07:51:00	-73.968095	40.768008	-73.956655	40.783762	1

1.4 Performance Metric

RMSE : Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Also, Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors.

So, RMSE becomes more useful when large errors are particularly undesirable. So, Root Mean Square value seems like a perfect choice for our problem at hand.

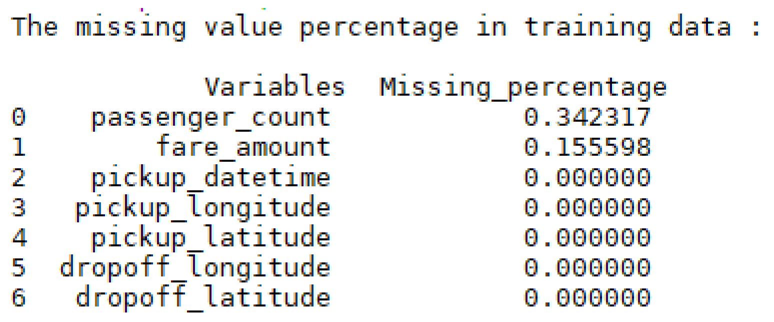
Chapter 2

Methodology

2.1.1 Data Preparation and Cleaning

2.1.1.1 Missing Value Analysis

One of the most common problems I have faced in Data Cleaning/Exploratory Data Analysis is handling the missing values. Firstly, there is no good way to deal with missing data. But still missing value analysis helps address several concerns caused by incomplete data. If cases with missing values are systematically different from cases without missing values, the results can be misleading. Also, missing data may reduce the precision of calculated statistics because there is less information than originally planned.



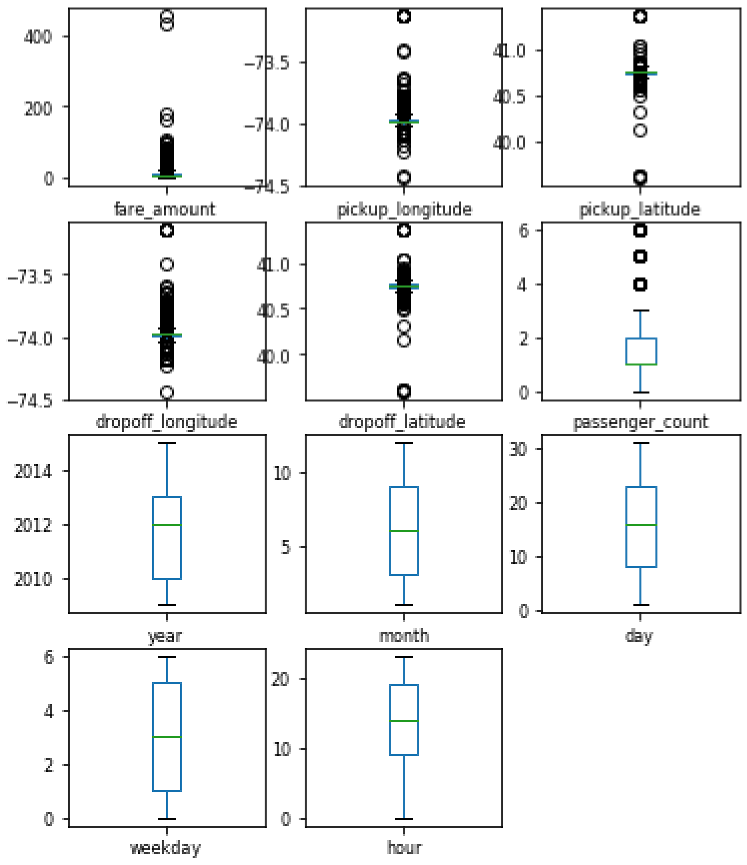
```
The missing value percentage in training data :

            Variables  Missing_percentage
0     passenger_count            0.342317
1         fare_amount            0.155598
2     pickup_datetime            0.000000
3    pickup_longitude            0.000000
4     pickup_latitude            0.000000
5   dropoff_longitude            0.000000
6    dropoff_latitude            0.000000
```

Another concern is that the assumptions behind many statistical procedures are based on complete cases, and missing values can complicate the theory required. So, In our data, there are plenty of missing values available in different variables. So, after computing the percentage of missing data that is available to us in the dataset, it accounts to around 1% of the data. It is also important to note that, the missing value has been calculated after removing the missing values within the target variable. We impute the missing values in other columns using Mean imputation, because that fits the best after trying various other imputation techniques like Mean, Median and Random value imputation.

2.1.1.2 Outliers Analysis

In statistics, an outlier is an observation point that is distant from other observations.In layman terms, we can say that an outlier is something which is separated/different from the crowd. Also, Outlier analysis is very important because they affect the mean and median which in turn affects the error (absolute and mean) in any data set. When we plot the error we might get big deviations if outliers are in the data set. In Box plots analysis of individual features, we can clearly observe from these boxplots that, not every feature contains outliers and many of them even have very few outliers.



Also, given the constraint that, we have only 16k data-points and after removing the outliers, the data gets decreased by almost 15%. So, dropping the outliers is probably not the best idea. Instead we will try to visualise and find out the outliers using box plots and will fill them with NA, that means we have created ‘missing values’ in place of outliers within the data. Now, we can treat these outliers like missing values and impute them using standard imputation techniques. In our case, we use Mean imputation to impute these missing values.

2.1.1.3 Feature Scaling

Normalization rescales the values into a range of [0,1]. This might be useful in some cases where all parameters need to have the same positive scale. However, the outliers from the data set are lost.

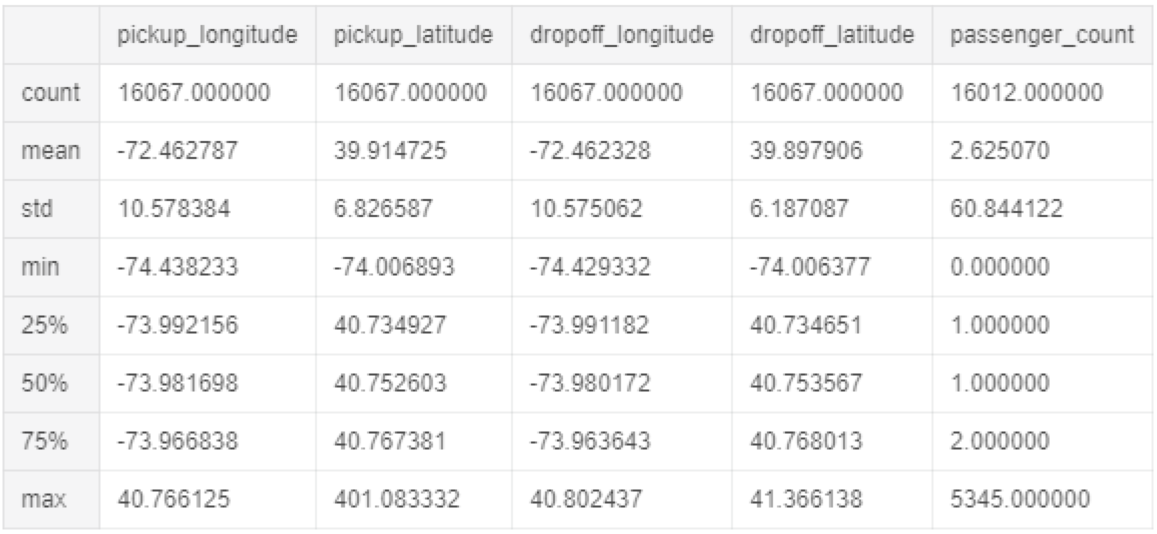
$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

In Linear Algebra, Normalization seems to refer to the dividing of a vector by its length.

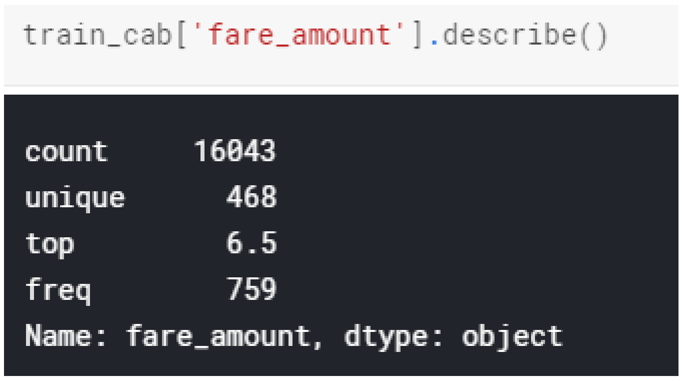
2.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the first step in our data analysis process. We do this by taking a broad look at patterns, trends, outliers, unexpected results and so on in our existing data, using visual and quantitative methods to get a sense of the story this tells. To start with this process, we will first have a look at univariate analysis like plotting Box plot and whiskers for individual features, Histogram plots, Bar plots and Kernel Density Estimation for the same for the same.

Description of Datasets:



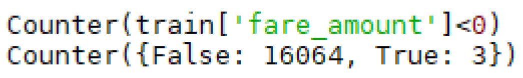
	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	16067.000000	16067.000000	16067.000000	16067.000000	16012.000000
mean	-72.462787	39.914725	-72.462328	39.897906	2.625070
std	10.578384	6.826587	10.575062	6.187087	60.844122
min	-74.438233	-74.006893	-74.429332	-74.006377	0.000000
25%	-73.992156	40.734927	-73.991182	40.734651	1.000000
50%	-73.981698	40.752603	-73.980172	40.753567	1.000000
75%	-73.966838	40.767381	-73.963643	40.768013	2.000000
max	40.766125	401.083332	40.802437	41.366138	5345.000000



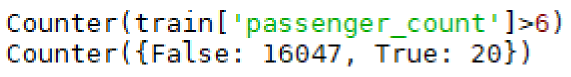
```
train_cab['fare_amount'].describe()
```

```
count     16043
unique      468
top         6.5
freq        759
Name: fare_amount, dtype: object
```

The columns 'fare_amount' and 'passenger_count' are having negative (-ve) values which need to be removed. Also Latitude and longitude values need to be in proper range.



```
Counter(train['fare_amount']<0)
Counter({False: 16064, True: 3})
```



```
Counter(train['passenger_count']>6)
Counter({False: 16047, True: 20})
```

How in a cab more than 6 passengers, can sit ?? Even 6 passengers are not possible. We will filter these excessive values in Outliers Analysis.

2.2.1 Data Visualisation

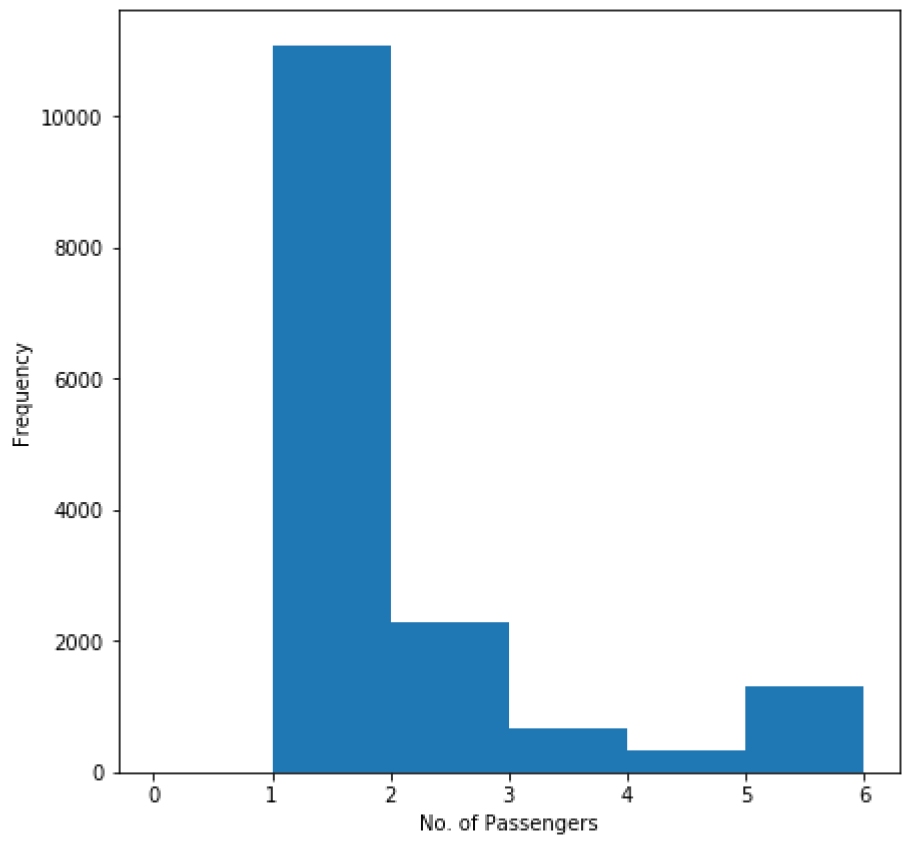
Data visualisation helps us to get better insights of the data. By visualising data, we can identify areas that need attention or improvement and also clarifies which factors influence fare of the cab and how the resources

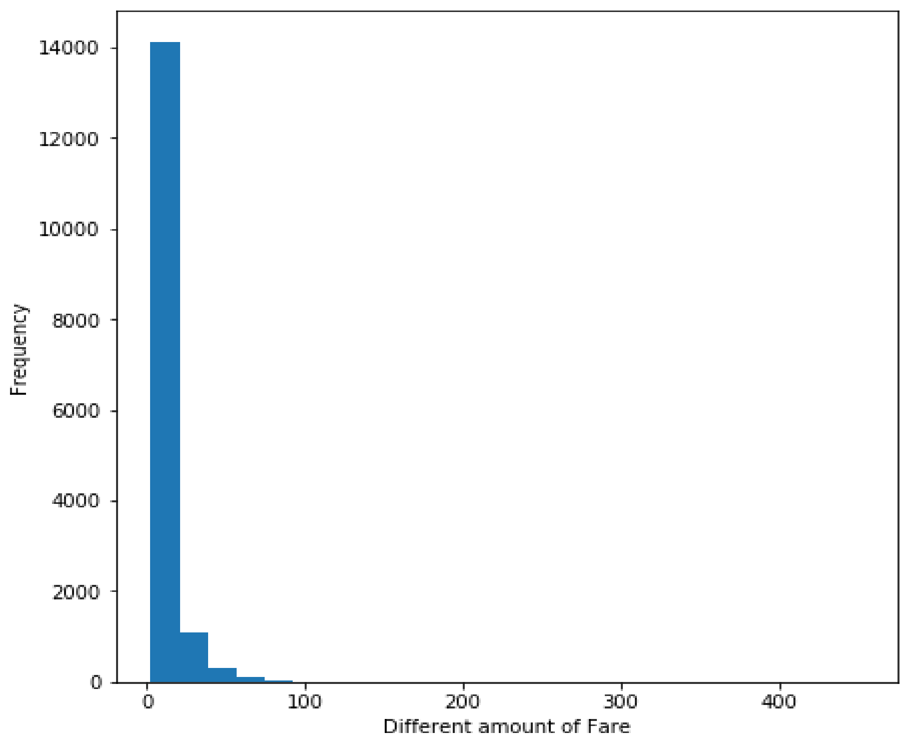
are used to determine it.

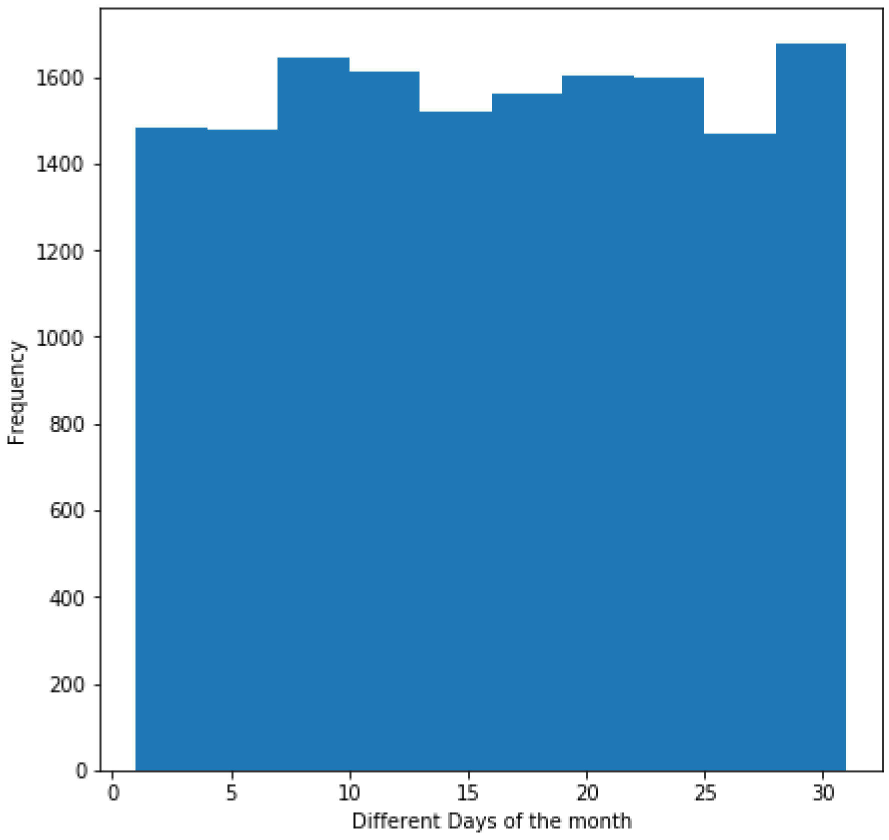
2.2.1.1 Univariate Analysis

Univariate analysis is the simplest form of data analysis where the data being analysed contains only one variable. Since it's a single variable it doesn’t deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it. So, Lets have a look at histogram plot, to identify the characteristic of the features and the data.

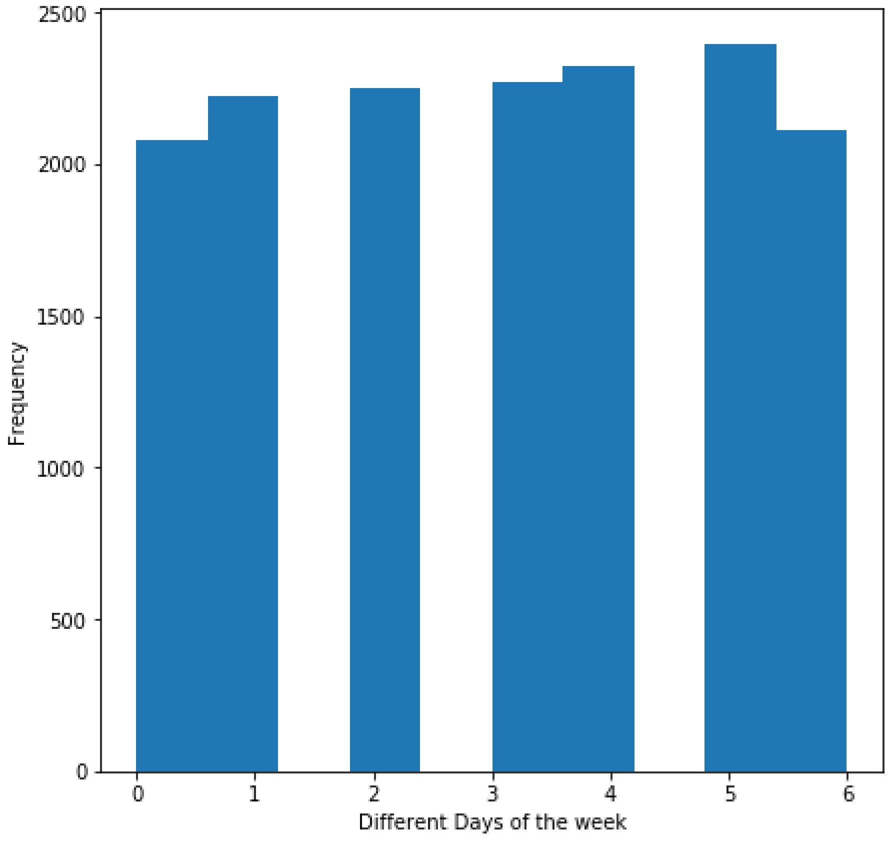
Histograms are constructed by binning the data and counting the number of observations in each bin. The objective of plotting Histogram plot is usually to visualise the shape of the distribution.The number of bins needs to be large enough to reveal interesting features and small enough not to be too noisy.



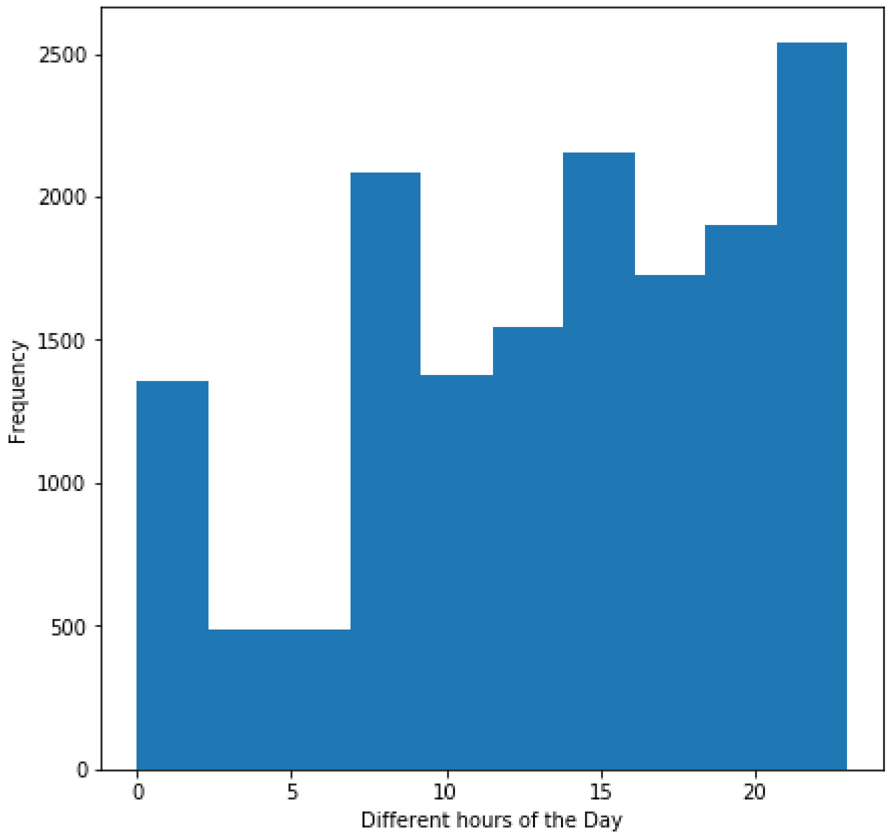




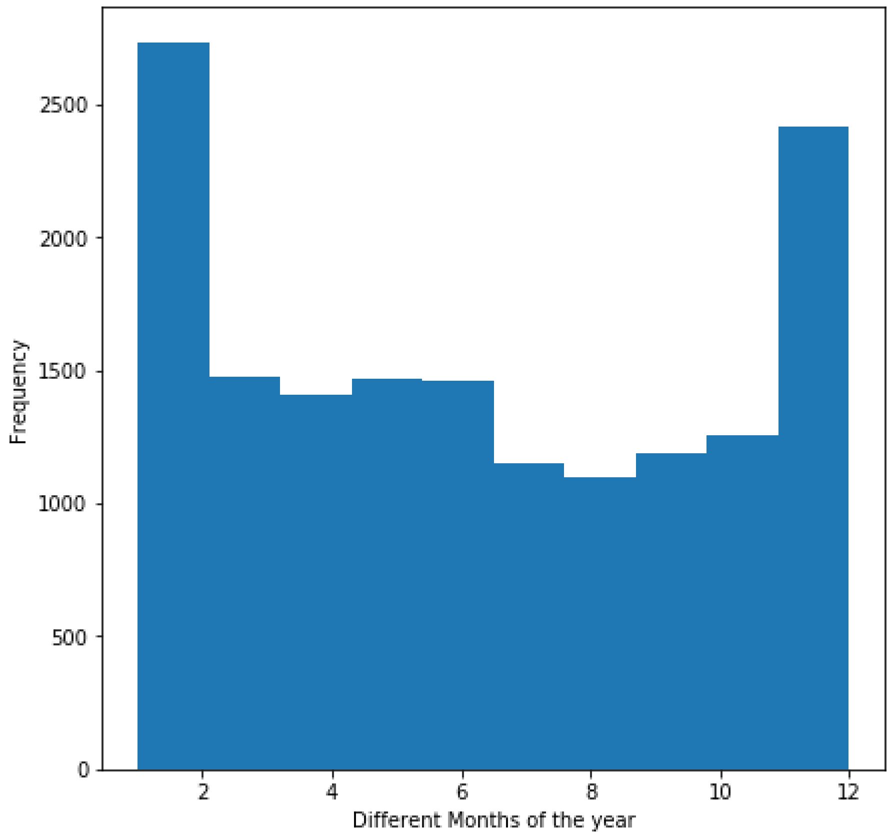
Frequency
0
200
400
600
800
1000
1200
1400
1600
0
5
10
15
20
25
30
Different Days of the month



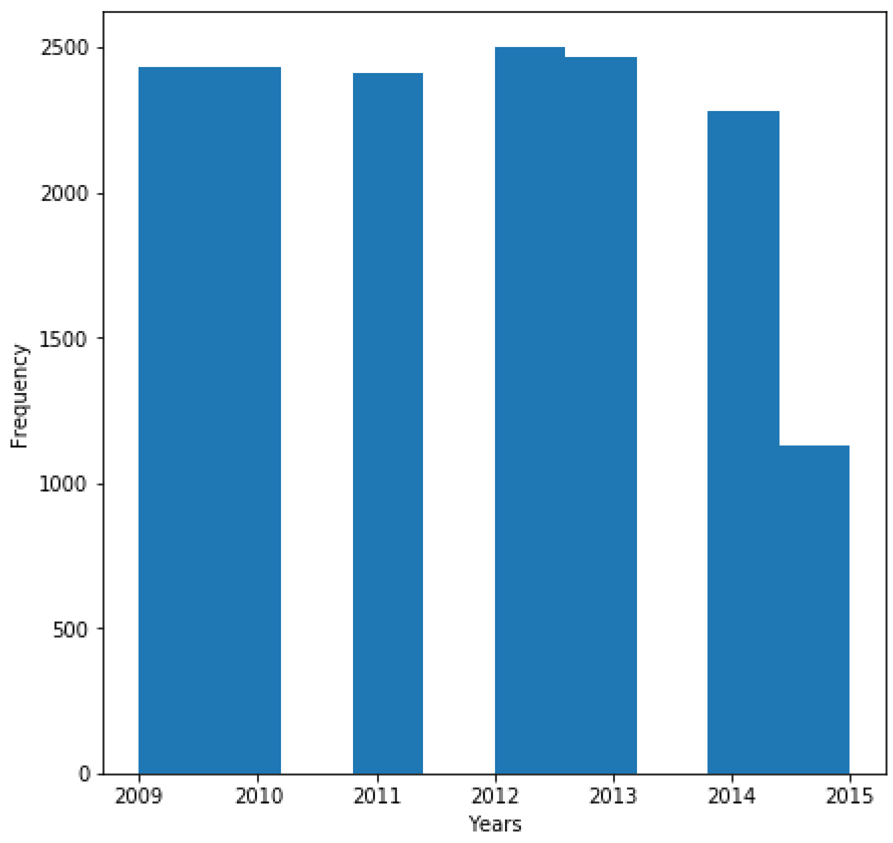
Frequency
0
500
1000
1500
2000
2500
0
1
2
3
4
5
6
Different Days of the week



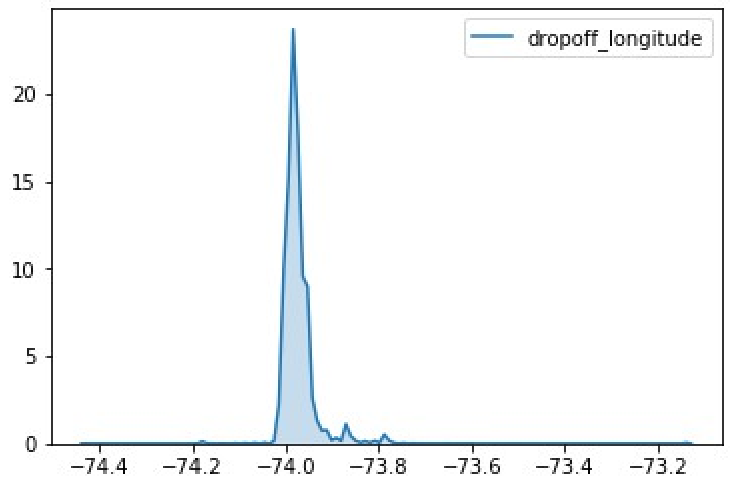
Frequency
2500
2000
1500
1000
500
0
0
5
10
15
20
Different hours of the Day

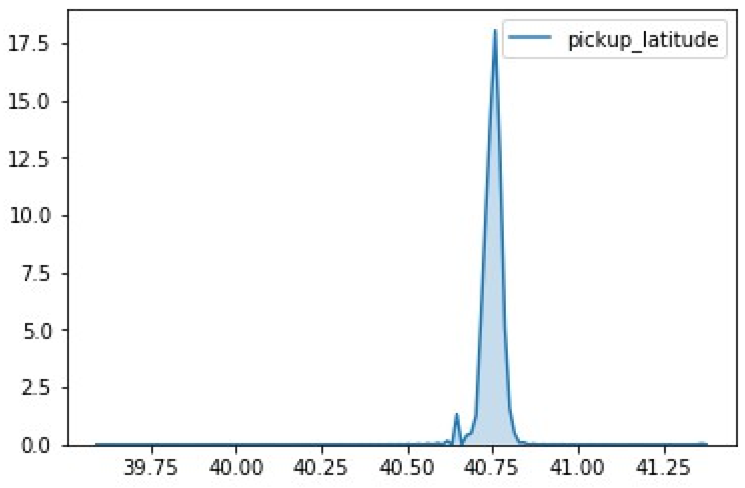


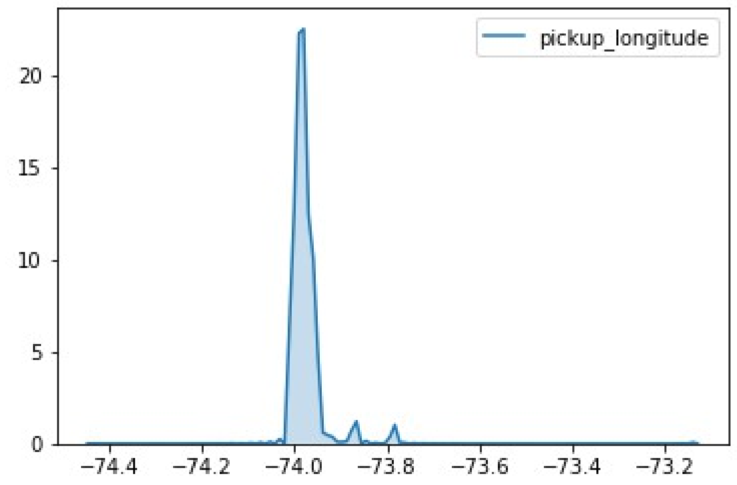
Frequency
2500
2000
1500
1000
500
0
2
4
6
8
10
12
Different Months of the year

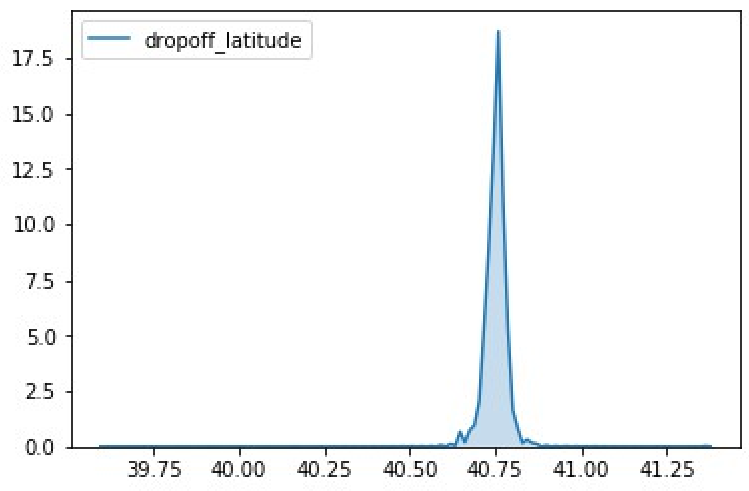


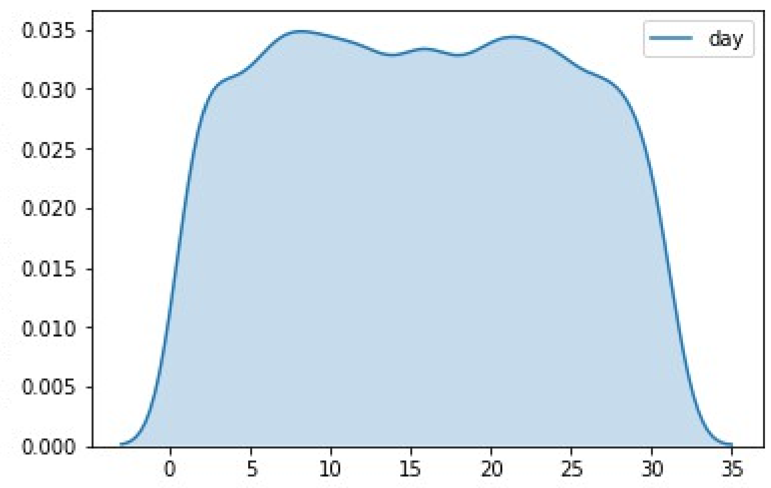
A Density Plot visualises the distribution of data over a continuous interval or time period. Density plots can be thought of as plots of smoothed histograms. An advantage Density Plots have over Histograms is that they're better at determining the distribution shape because they're not affected by the number of bins used.



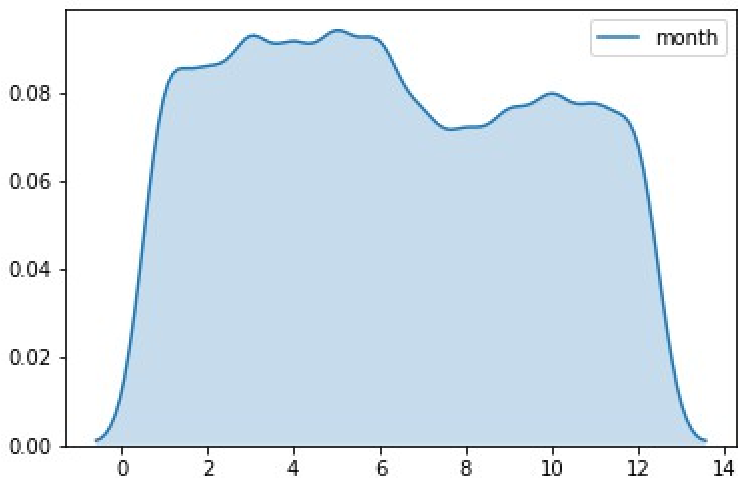

pickup_latitude
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
39.75
40.00
40.25
40.50
40.75
41.00
41.25


pickup_longitude
20
15
10
5
0
−74.4
−74.2
−74.0
−73.8
−73.6
−73.4
−73.2

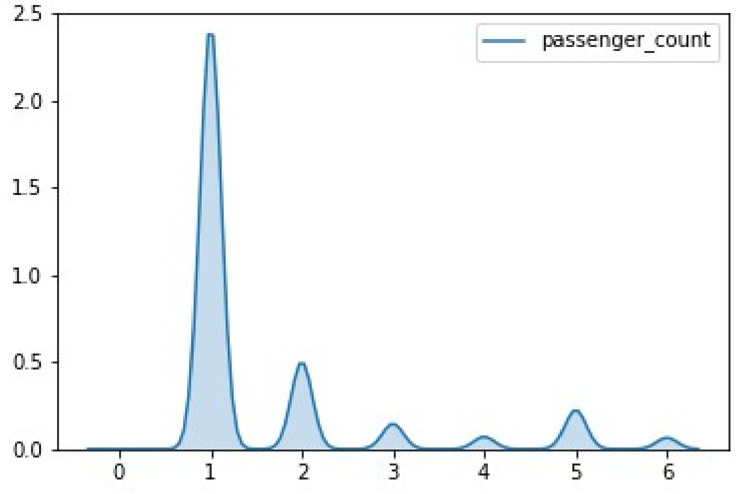

dropoff_latitude
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
39.75
40.00
40.25
40.50
40.75
41.00
41.25



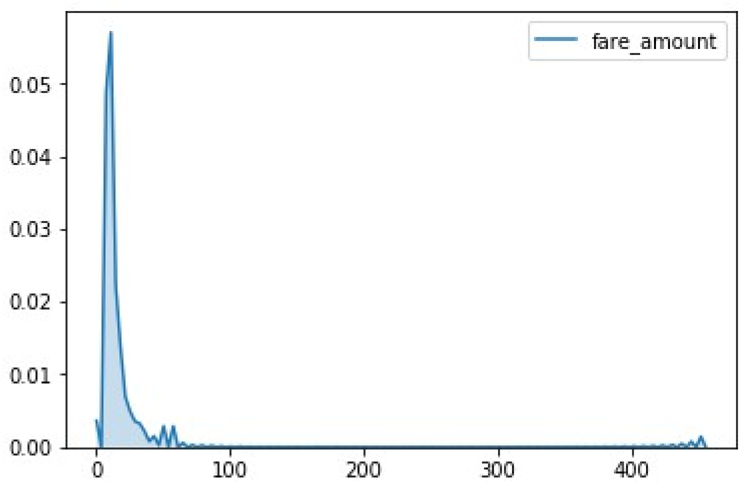
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
5
10
15
20
25
30
35
day



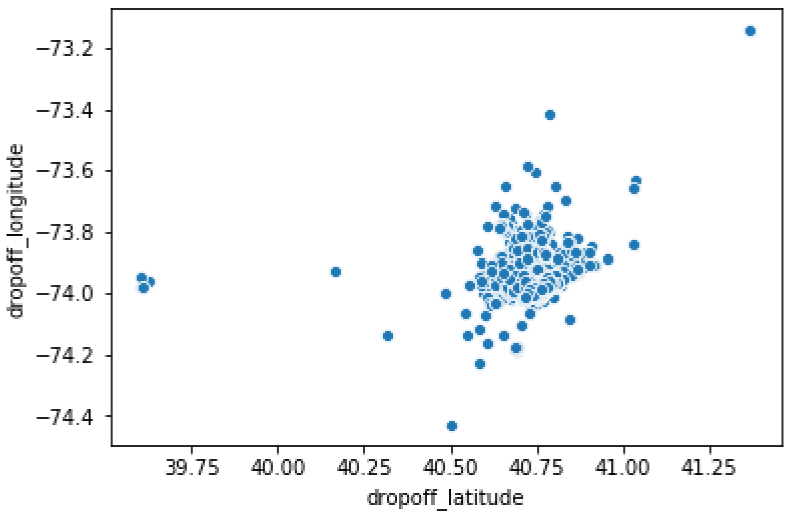
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8
10
12
14
month

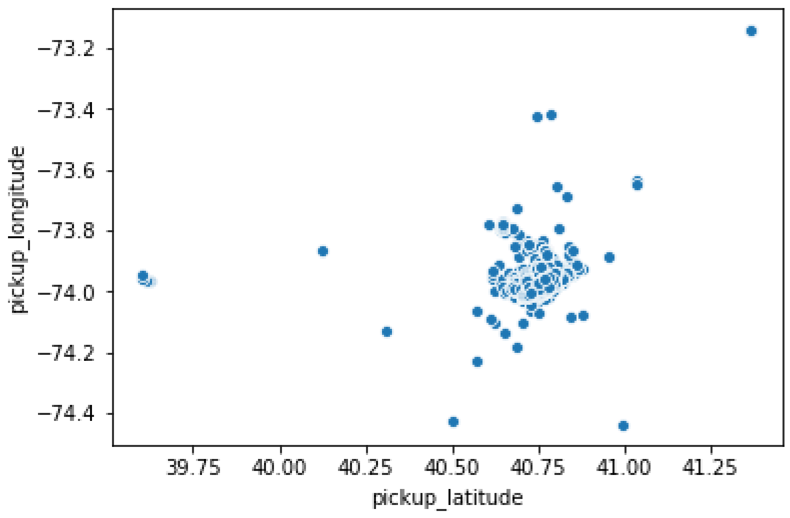


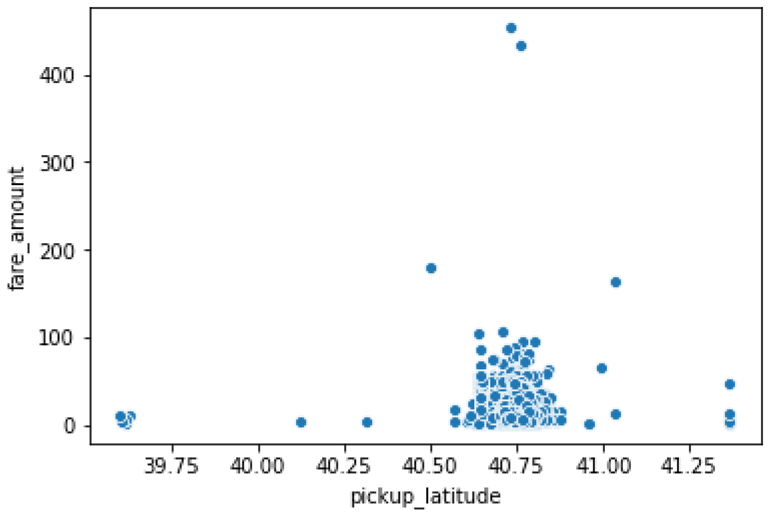
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
6
passenger_count



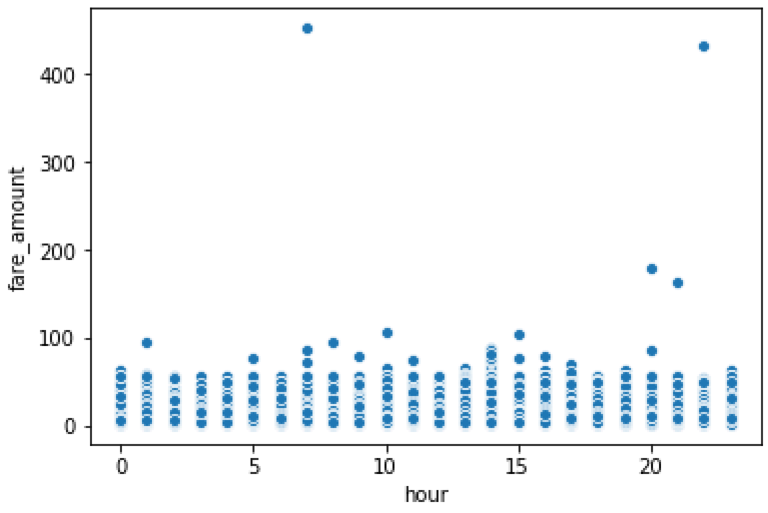
2.1.2.2 Bivariate Analysis



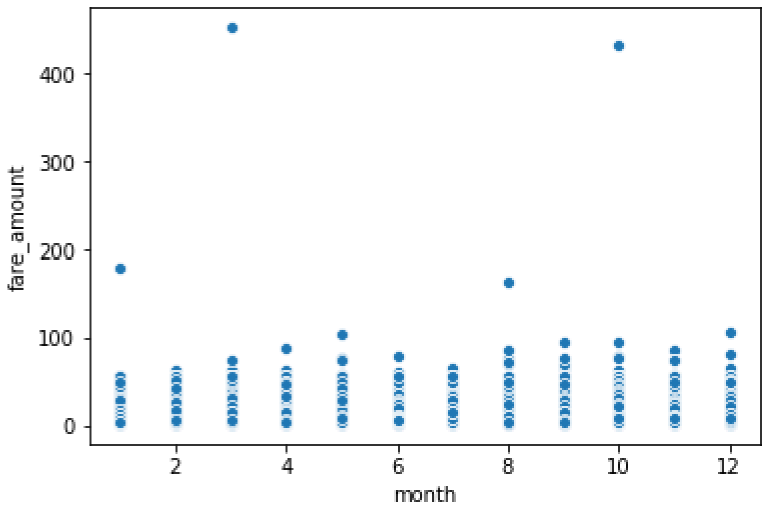




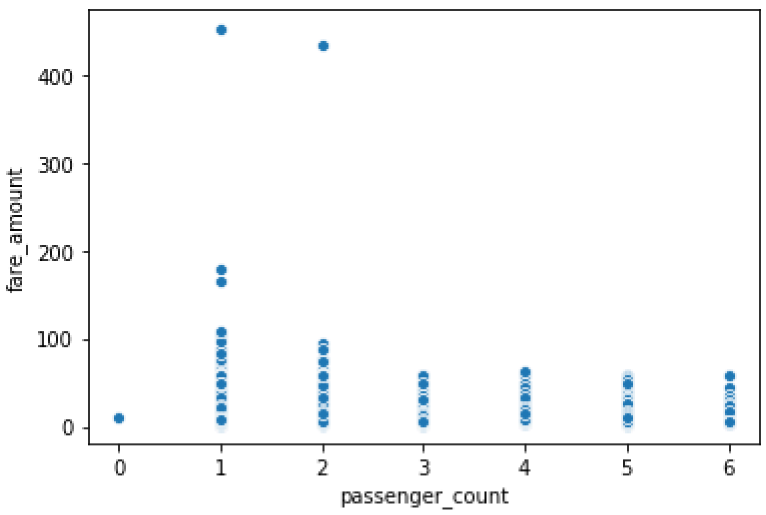
fare_amount
pickup_latitude
0
100
200
300
400
39.75
40.00
40.25
40.50
40.75
41.00
41.25



fare_amount
hour
0
100
200
300
400
0
5
10
15
20



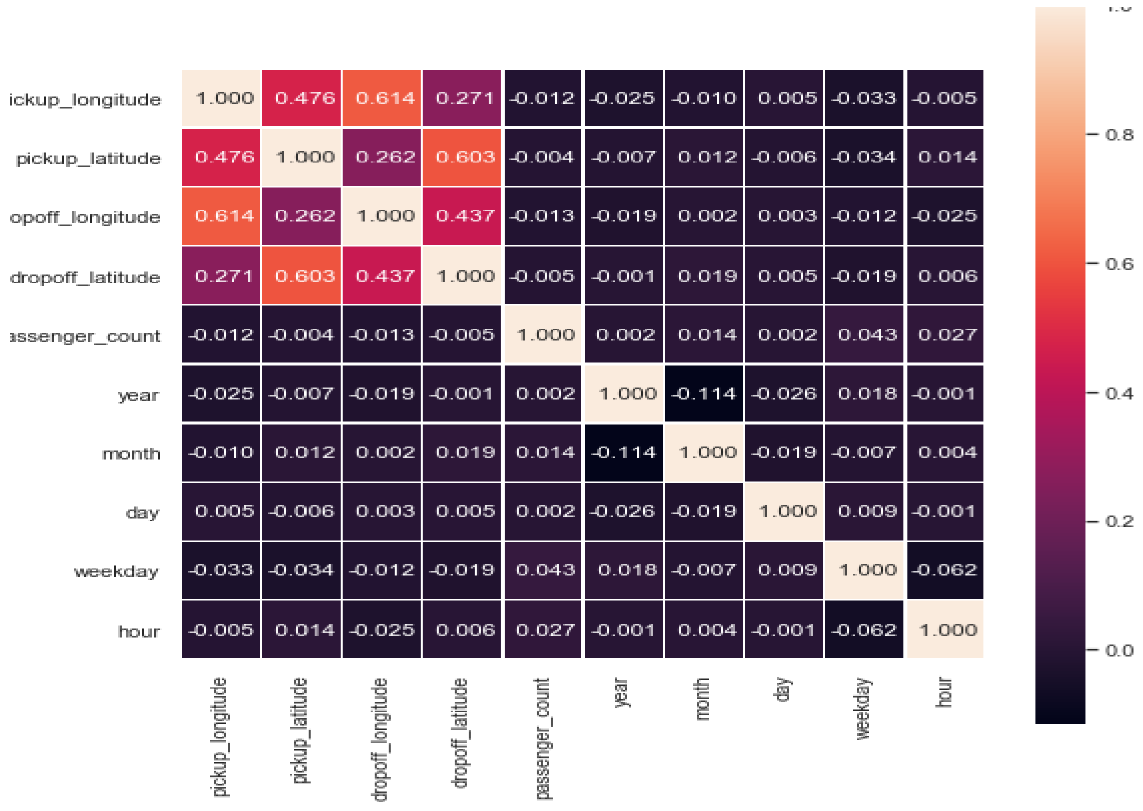
fare_amount
month
0
100
200
300
400
2
4
6
8
10
12



2.1.2.3 Feature Selection

Correlation Analysis:

If the values of one column to other column are similar then it is said to be collinear. Therefore between predictor variables there should be less collinearity as compared to the collinearity among the predictors and variable.



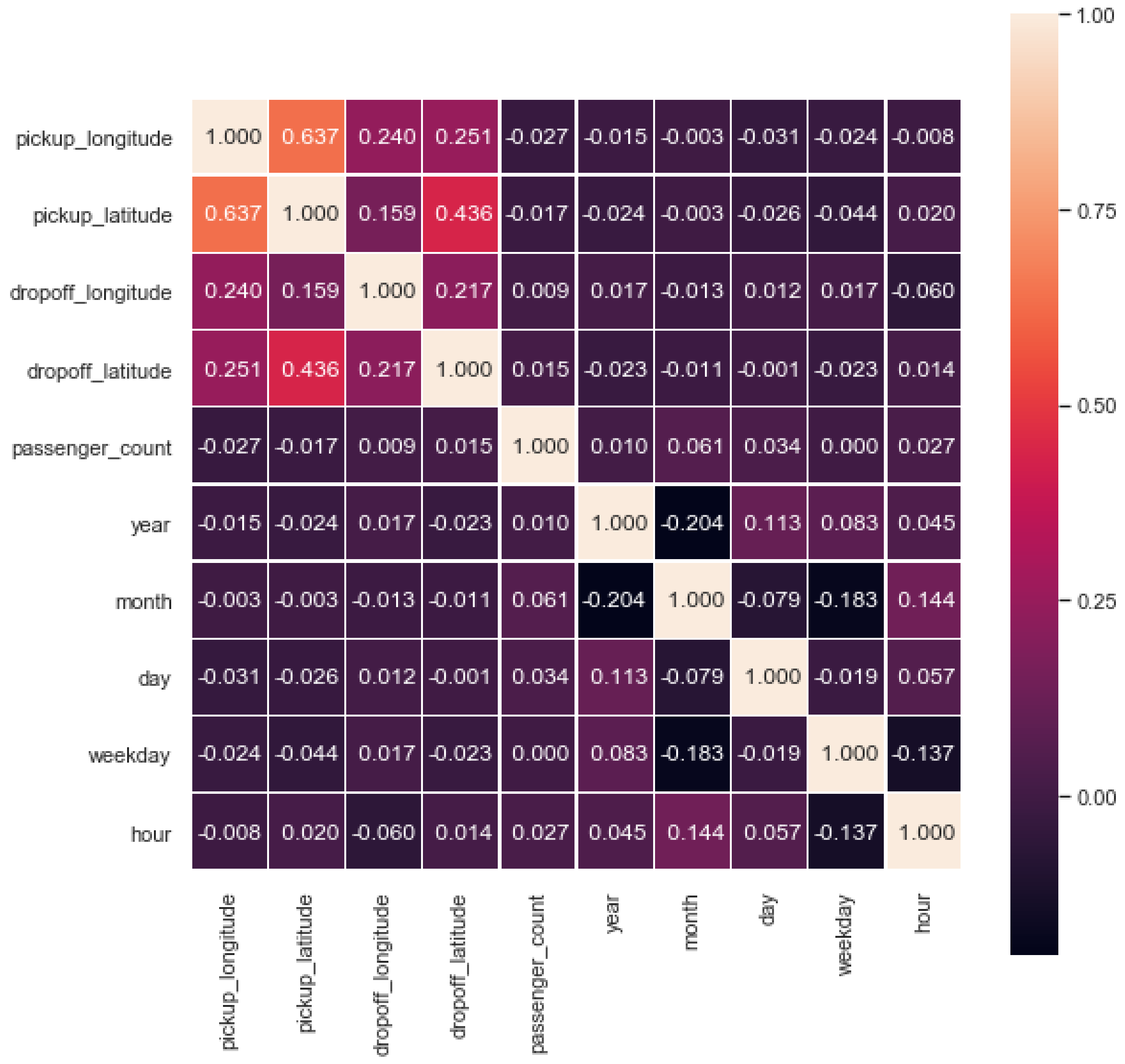
	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	year	month	day	weekday	hour
ickup_longitude	1.000	0.476	0.614	0.271	-0.012	-0.025	-0.010	0.005	-0.033	-0.005
pickup_latitude	0.476	1.000	0.262	0.603	-0.004	-0.007	0.012	-0.006	-0.034	0.014
opoff_longitude	0.614	0.262	1.000	0.437	-0.013	-0.019	0.002	0.003	-0.012	-0.025
dropoff_latitude	0.271	0.603	0.437	1.000	-0.005	-0.001	0.019	0.005	-0.019	0.006
assenger_count	-0.012	-0.004	-0.013	-0.005	1.000	0.002	0.014	0.002	0.043	0.027
year	-0.025	-0.007	-0.019	-0.001	0.002	1.000	-0.114	-0.026	0.018	-0.001
month	-0.010	0.012	0.002	0.019	0.014	-0.114	1.000	-0.019	-0.007	0.004
day	0.005	-0.006	0.003	0.005	0.002	-0.026	-0.019	1.000	0.009	-0.001
weekday	-0.033	-0.034	-0.012	-0.019	0.043	0.018	-0.007	0.009	1.000	-0.062
hour	-0.005	0.014	-0.025	0.006	0.027	-0.001	0.004	-0.001	-0.062	1.000

Fig: Collinearity test in train data

The value for collinearity is between -1 to 1. So, any value close to -1/1 will result in high collinearity.

It seems all right in our train and test data the situations nothing is more than 0.4 is positive direction and nothing is less than -0.1 is negative direction.

Therefore the datasets are free from collinearity problem.

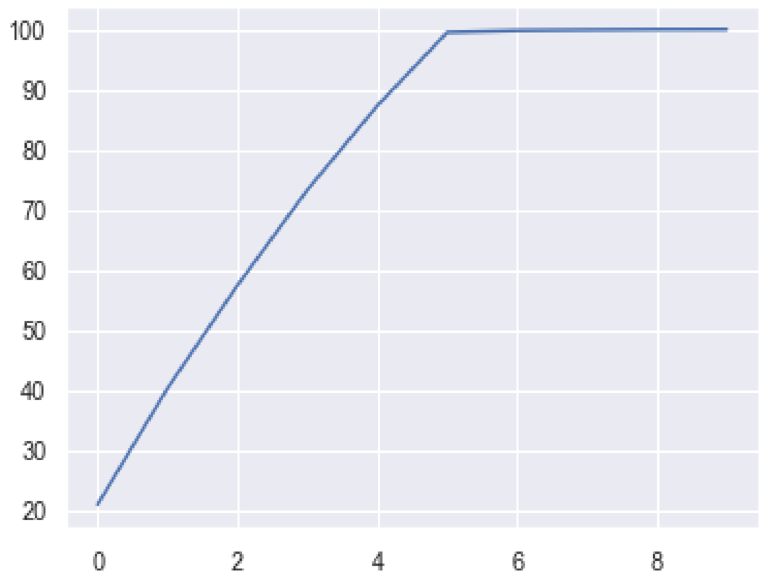


	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	year	month	day	weekday	hour
pickup_longitude	1.000	0.637	0.240	0.251	-0.027	-0.015	-0.003	-0.031	-0.024	-0.008
pickup_latitude	0.637	1.000	0.159	0.436	-0.017	-0.024	-0.003	-0.026	-0.044	0.020
dropoff_longitude	0.240	0.159	1.000	0.217	0.009	0.017	-0.013	0.012	0.017	-0.060
dropoff_latitude	0.251	0.436	0.217	1.000	0.015	-0.023	-0.011	-0.001	-0.023	0.014
passenger_count	-0.027	-0.017	0.009	0.015	1.000	0.010	0.061	0.034	0.000	0.027
year	-0.015	-0.024	0.017	-0.023	0.010	1.000	-0.204	0.113	0.083	0.045
month	-0.003	-0.003	-0.013	-0.011	0.061	-0.204	1.000	-0.079	-0.183	0.144
day	-0.031	-0.026	0.012	-0.001	0.034	0.113	-0.079	1.000	-0.019	0.057
weekday	-0.024	-0.044	0.017	-0.023	0.000	0.083	-0.183	-0.019	1.000	-0.137
hour	-0.008	0.020	-0.060	0.014	0.027	0.045	0.144	0.057	-0.137	1.000

Dimension Reduction :

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that. Here, we do this in two steps : Firstly, we find and remove the correlated features and then we use a more advanced technique for dimensionality reduction called PCA .While doing this, we first plot a cumulative distribution function plot to observe how much percentage of variance is explained by how many variables (Principle Components).

The CDF plot for the same is plotted below :



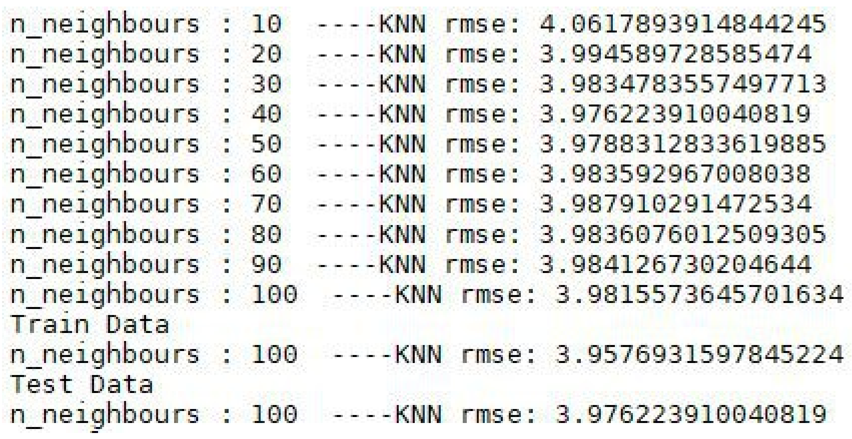
It is very clear from the above CDF plot for ‘Variance Explained’ Vs ‘Principle components’ , that almost 99%+ variance is explained by just 5 variables (Principle components). We, can imagine how powerful PCA is, It just shrank down our feature space to just 5 from a total of 7 features. So, we will keep only 5 principle components in the data and will perform modeling on it.

2.2 Modeling

We always start our model building from the most simplest to more complex. Therefore we start with KNN Regressor.

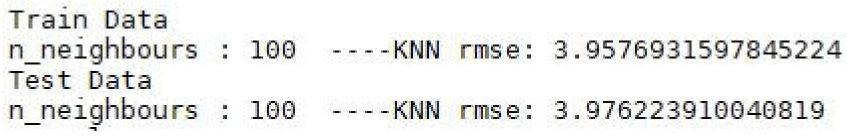
2.2.1 KNN Regression

KNN regression is one of the simplest algorithm in the whole of Machine learning. It gives a weighted average of the regression function in a local space (k nearest points to a given point). So, we first try to implement and fit KNN regression to our Data and got following results after tuning the hyper-parameter k :



```
n_neighbours : 10  ----KNN rmse: 4.0617893914844245
n_neighbours : 20  ----KNN rmse: 3.994589728585474
n_neighbours : 30  ----KNN rmse: 3.9834783557497713
n_neighbours : 40  ----KNN rmse: 3.976223910040819
n_neighbours : 50  ----KNN rmse: 3.9788312833619885
n_neighbours : 60  ----KNN rmse: 3.983592967008038
n_neighbours : 70  ----KNN rmse: 3.987910291472534
n_neighbours : 80  ----KNN rmse: 3.9836076012509305
n_neighbours : 90  ----KNN rmse: 3.984126730204644
n_neighbours : 100  ----KNN rmse: 3.9815573645701634
Train Data
n_neighbours : 100  ----KNN rmse: 3.9576931597845224
Test Data
n_neighbours : 100  ----KNN rmse: 3.976223910040819
```

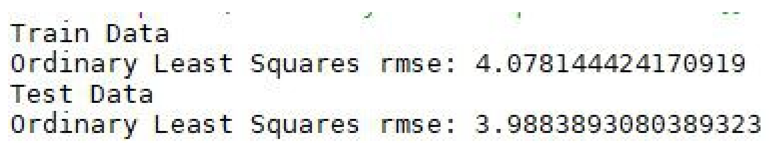
So, n_neighbors =100 seems to have best RMSE value, Lets fit and predict it on our data :



```
Train Data
n_neighbours : 100  ----KNN rmse: 3.9576931597845224
Test Data
n_neighbours : 100  ----KNN rmse: 3.976223910040819
```

2.2.2 Ordinary Least Squares

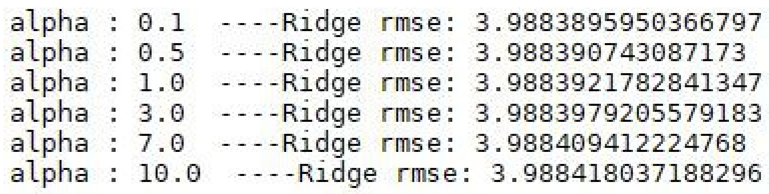
Now we will try to implement Multiple Linear Regression algorithm using Ordinary Least Squares, the simplest of all.Ordinary least squares (OLS) minimises the squared distances between the observed and the predicted dependent variable.



```
Train Data
Ordinary Least Squares rmse: 4.078144424170919
Test Data
Ordinary Least Squares rmse: 3.9883893080389323
```

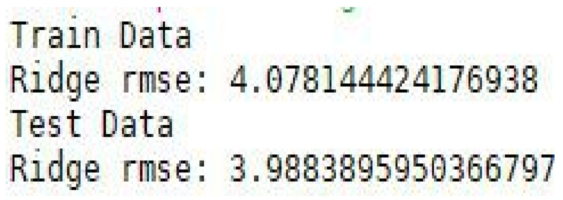
2.2.3 Ridge Regression

Ridge Regression essentially is an instance of Linear Regression with regularisation. Ridge regression is that it enforces the β coefficients to be lower, but it does not enforce them to be zero.



```
alpha : 0.1  ----Ridge rmse: 3.9883895950366797
alpha : 0.5  ----Ridge rmse: 3.988390743087173
alpha : 1.0  ----Ridge rmse: 3.9883921782841347
alpha : 3.0  ----Ridge rmse: 3.9883979205579183
alpha : 7.0  ----Ridge rmse: 3.988409412224768
alpha : 10.0  ----Ridge rmse: 3.988418037188296
```

That is, it will not get rid of irrelevant features but rather minimise their impact on the trained model. So,after we implement Ridge Regression on our data, we get the following results :

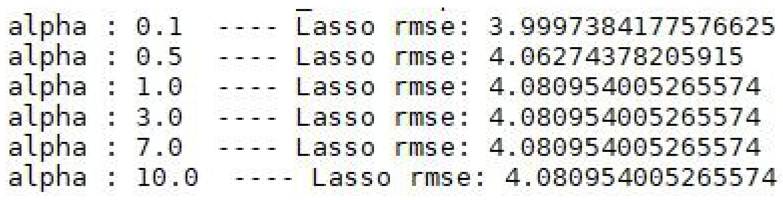


```
Train Data
Ridge rmse: 4.078144424176938
Test Data
Ridge rmse: 3.9883895950366797
```

So, we can see from the above results that, our model gives a RMSE (Root Mean Square Value) of 4.07 for the train data and a RMSE value of 3.98 for the test data. Looking at the train and test error, we can say that the model doesn’t fit at all and that might be because of the regularisation term involve in the cost function.

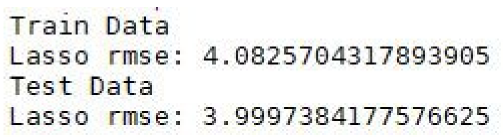
2.2.4 Lasso Regression

Least absolute shrinkage and selection operator, abbreviated as LASSO, is an Linear Regression technique which also performs regularisation on variables in consideration. It sets the coefficients to zero thus reducing the errors completely. That is, It will get rid of irrelevant features completely.



```
alpha : 0.1  ---- Lasso rmse: 3.9997384177576625
alpha : 0.5  ---- Lasso rmse: 4.06274378205915
alpha : 1.0  ---- Lasso rmse: 4.080954005265574
alpha : 3.0  ---- Lasso rmse: 4.080954005265574
alpha : 7.0  ---- Lasso rmse: 4.080954005265574
alpha : 10.0  ---- Lasso rmse: 4.080954005265574
```

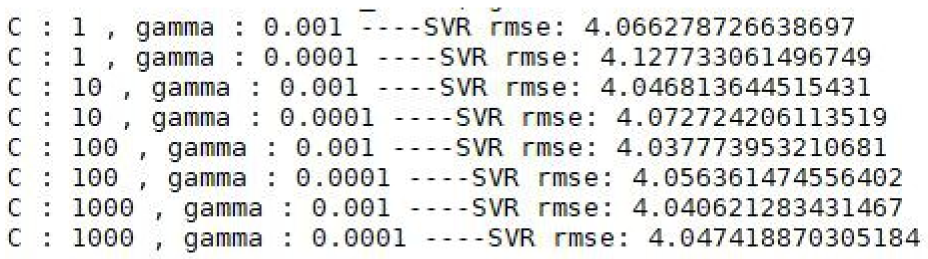
So, after we implement Lasso Regression on our data, we get the following results :



```
Train Data
Lasso rmse: 4.0825704317893905
Test Data
Lasso rmse: 3.9997384177576625
```

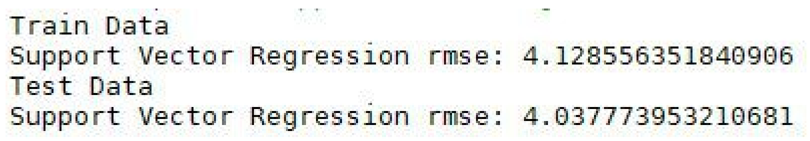
2.2.5 Support Vector Regression

Support Vector Machine can be applied not only to classification problems but also to the case of regression. Still it contains all the main features that characterise maximum margin algorithm: a non-linear function is leaned by linear learning machine mapping into high dimensional kernel induced feature space.



```
C : 1 , gamma : 0.001 ----SVR rmse: 4.066278726638697
C : 1 , gamma : 0.0001 ----SVR rmse: 4.127733061496749
C : 10 , gamma : 0.001 ----SVR rmse: 4.046813644515431
C : 10 , gamma : 0.0001 ----SVR rmse: 4.072724206113519
C : 100 , gamma : 0.001 ----SVR rmse: 4.037773953210681
C : 100 , gamma : 0.0001 ----SVR rmse: 4.056361474556402
C : 1000 , gamma : 0.001 ----SVR rmse: 4.040621283431467
C : 1000 , gamma : 0.0001 ----SVR rmse: 4.047418870305184
```

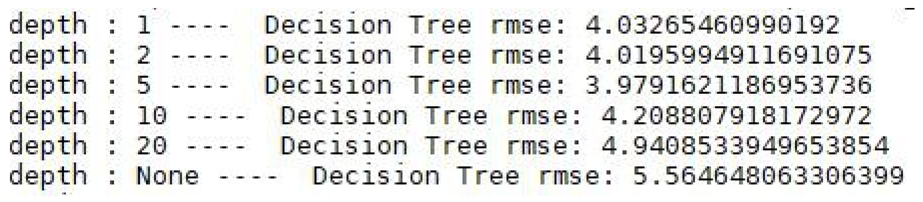
So, after we implement Support Vector Regression on our data, we get the following results :



```
Train Data
Support Vector Regression rmse: 4.128556351840906
Test Data
Support Vector Regression rmse: 4.037773953210681
```

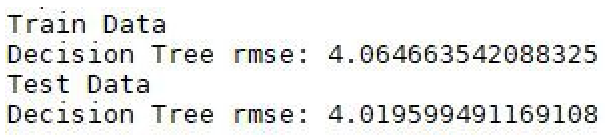
2.2.6 Decision Tree Regression

Decision tree builds regression , models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.



```
depth : 1 ----  Decision Tree rmse: 4.03265460990192
depth : 2 ----  Decision Tree rmse: 4.0195994911691075
depth : 5 ----  Decision Tree rmse: 3.9791621186953736
depth : 10 ----  Decision Tree rmse: 4.208807918172972
depth : 20 ----  Decision Tree rmse: 4.9408533949653854
depth : None ----  Decision Tree rmse: 5.564648063306399
```

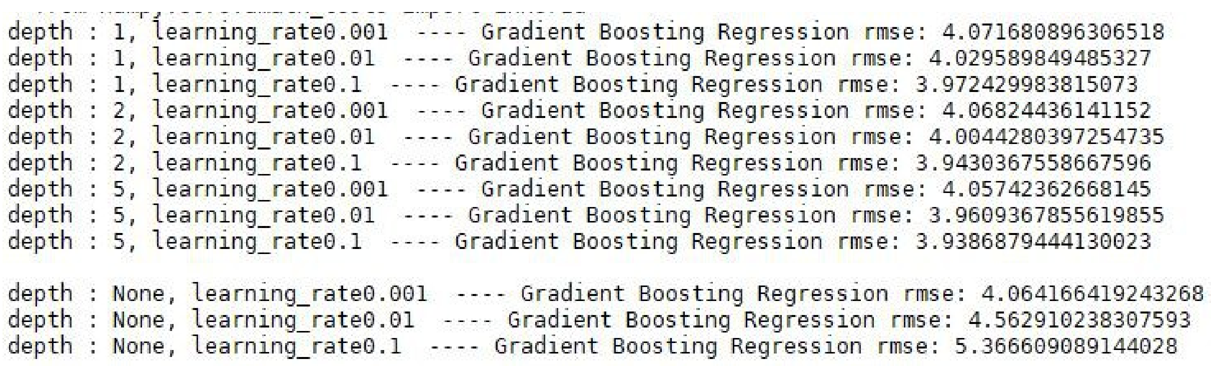
The final result is a tree with decision nodes and leaf nodes. So, after we implement Decision Tree Regression on our data, we get the following results :



```
Train Data
Decision Tree rmse: 4.064663542088325
Test Data
Decision Tree rmse: 4.019599491169108
```

2.2.7 Gradient Boosting Decision Tree

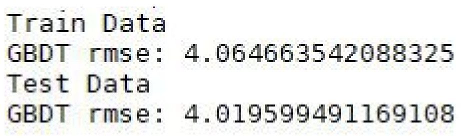
Regression Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalises them by allowing optimisation of an arbitrary differentiable loss function.



```
depth : 1, learning_rate0.001  ---- Gradient Boosting Regression rmse: 4.071680896306518
depth : 1, learning_rate0.01  ---- Gradient Boosting Regression rmse: 4.029589849485327
depth : 1, learning_rate0.1  ---- Gradient Boosting Regression rmse: 3.972429983815073
depth : 2, learning_rate0.001  ---- Gradient Boosting Regression rmse: 4.06824436141152
depth : 2, learning_rate0.01  ---- Gradient Boosting Regression rmse: 4.0044280397254735
depth : 2, learning_rate0.1  ---- Gradient Boosting Regression rmse: 3.9430367558667596
depth : 5, learning_rate0.001  ---- Gradient Boosting Regression rmse: 4.05742362668145
depth : 5, learning_rate0.01  ---- Gradient Boosting Regression rmse: 3.9609367855619855
depth : 5, learning_rate0.1  ---- Gradient Boosting Regression rmse: 3.9386879444130023

depth : None, learning_rate0.001  ---- Gradient Boosting Regression rmse: 4.064166419243268
depth : None, learning_rate0.01  ---- Gradient Boosting Regression rmse: 4.562910238307593
depth : None, learning_rate0.1  ---- Gradient Boosting Regression rmse: 5.366609089144028
```

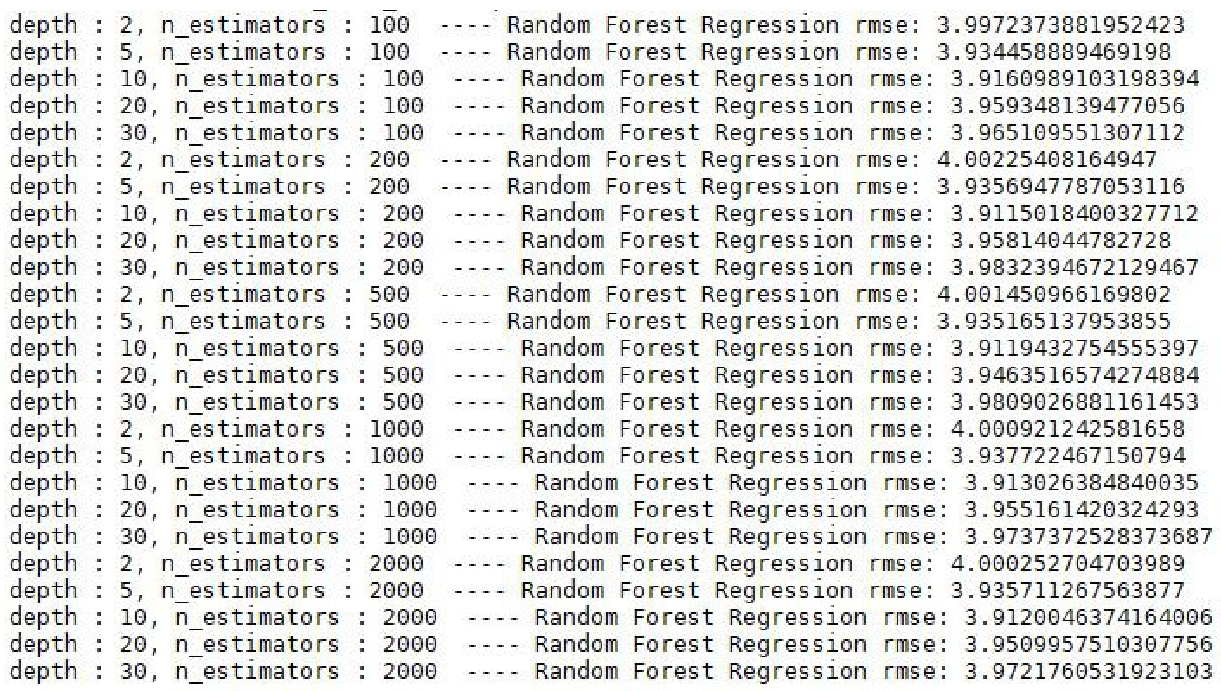
So, after we implement Gradient Boosting Decision Trees on our data, we get the following results :



```
Train Data
GBDT rmse: 4.064663542088325
Test Data
GBDT rmse: 4.019599491169108
```

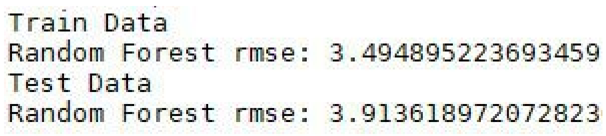
2.2.8 Random Forest Regression

Random Forest Regression or Regression Trees are know to be very unstable, in other words, a small change in our data may drastically change your model. The Random Forest uses this instability as an advantage through bagging resulting on a very stable model.



```
depth : 2, n_estimators : 100  ---- Random Forest Regression rmse: 3.9972373881952423
depth : 5, n_estimators : 100  ---- Random Forest Regression rmse: 3.934458889469198
depth : 10, n_estimators : 100  ---- Random Forest Regression rmse: 3.9160989103198394
depth : 20, n_estimators : 100  ---- Random Forest Regression rmse: 3.959348139477056
depth : 30, n_estimators : 100  ---- Random Forest Regression rmse: 3.965109551307112
depth : 2, n_estimators : 200  ---- Random Forest Regression rmse: 4.00225408164947
depth : 5, n_estimators : 200  ---- Random Forest Regression rmse: 3.9356947787053116
depth : 10, n_estimators : 200  ---- Random Forest Regression rmse: 3.9115018400327712
depth : 20, n_estimators : 200  ---- Random Forest Regression rmse: 3.95814044782728
depth : 30, n_estimators : 200  ---- Random Forest Regression rmse: 3.9832394672129467
depth : 2, n_estimators : 500  ---- Random Forest Regression rmse: 4.001450966169802
depth : 5, n_estimators : 500  ---- Random Forest Regression rmse: 3.935165137953855
depth : 10, n_estimators : 500  ---- Random Forest Regression rmse: 3.9119432754555397
depth : 20, n_estimators : 500  ---- Random Forest Regression rmse: 3.9463516574274884
depth : 30, n_estimators : 500  ---- Random Forest Regression rmse: 3.9809026881161453
depth : 2, n_estimators : 1000  ---- Random Forest Regression rmse: 4.000921242581658
depth : 5, n_estimators : 1000  ---- Random Forest Regression rmse: 3.937722467150794
depth : 10, n_estimators : 1000  ---- Random Forest Regression rmse: 3.913026384840035
depth : 20, n_estimators : 1000  ---- Random Forest Regression rmse: 3.955161420324293
depth : 30, n_estimators : 1000  ---- Random Forest Regression rmse: 3.9737372528373687
depth : 2, n_estimators : 2000  ---- Random Forest Regression rmse: 4.000252704703989
depth : 5, n_estimators : 2000  ---- Random Forest Regression rmse: 3.935711267563877
depth : 10, n_estimators : 2000  ---- Random Forest Regression rmse: 3.9120046374164006
depth : 20, n_estimators : 2000  ---- Random Forest Regression rmse: 3.9509957510307756
depth : 30, n_estimators : 2000  ---- Random Forest Regression rmse: 3.9721760531923103
```

So, after we implement Random Forest Regression or Regression Trees on our data, we get the following results :



```
Train Data
Random Forest rmse: 3.494895223693459
Test Data
Random Forest rmse: 3.913618972072823
```

RESULTS IN R :

```
> #### KNN
> #Develop Model on training data
> fit_LR = knnreg(fare_amount ~ ., data = X_train.data)
> #Lets predict for testing data
> pred_LR_test = predict(fit_LR,X_test.data)
> # Results
> print(postResample(pred = pred_LR_test, obs =y_test$fare_amount))
     RMSE  Rsquared       MAE 
0.1661820 0.3416006 0.1212201 

> ###### Multiple Linear Regression
> #Develop Model on training data
> set.seed(100)
> #Develop Model on training data
> fit_LR = lm(fare_amount ~ ., data = X_train.data)
> #Lets predict for testing data
> pred_LR_test = predict(fit_LR,X_test.data)
> # Results
> print(postResample(pred = pred_LR_test, obs =y_test$fare_amount))
     RMSE  Rsquared       MAE 
0.1976116 0.0454827 0.1522892 
```

```
> ###### SVM
> #Develop Model on training data
> fit_LR = svm(fare_amount~ ., data = train, scale = FALSE,type = "eps")
> #Lets predict for testing data
> pred_LR_test = predict(fit_LR,X_test)
> # Results X_test
> print(postResample(pred = pred_LR_test, obs =y_test$fare_amount))
     RMSE  Rsquared       MAE 
0.1696942 0.3052170 0.1246571 

> ###### Decision Tree
> 
> #Develop Model on training data
> fit_DT = rpart(fare_amount ~., data = X_train.data, method = 'anova')
> pred_DT_test = predict(fit_DT,X_test.data)
> # Results
> print(postResample(pred = pred_DT_test, obs = y_test$fare_amount))
     RMSE  Rsquared       MAE 
0.1707359 0.2870093 0.1317889 

> ###### Random Forest
> #Develop Model on training data
> fit_DT = randomForest(fare_amount ~., data = train)
> pred_DT_test = predict(fit_DT,X_test)
> # Results
> print(postResample(pred = pred_DT_test, obs = y_test$fare_amount))
     RMSE  Rsquared       MAE 
0.1706328 0.3662941 0.1308497 

> #Develop Gradient tboosting Model
> fit_GBDT = gbm(fare_amount ~., data = X_train.data, n.trees = 500, 
interaction.depth = 2)
Distribution not specified, assuming gaussian ...
> #Lets predict for testing data
> pred_GBDT_test = predict(fit_GBDT,X_test.data, n.trees = 500)
> # For testing data
> print(postResample(pred = pred_GBDT_test, obs = y_test$fare_amount))
     RMSE  Rsquared       MAE 
0.1560855 0.4043815 0.1155628 
```

Chapter 3

Conclusion 3.1

Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance

2. Interpretability

3. Computational Efficiency

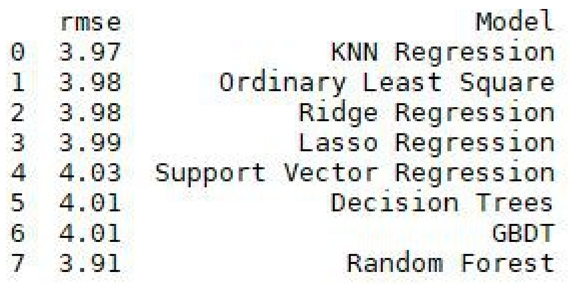
In our case of Cab fare , the latter two, Interpretability and Computation Efficiency, do not hold much significance. Therefore we will use Predictive performance as the criteria to compare and evaluate models. Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

3.1.1 Root Mean Square Value

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Also, Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. So, RMSE becomes more useful when large errors are particularly undesirable. So, Root Mean Square value seems like a perfect choice for our problem at hand.

3.2 Model Selection

We saw that both models Random Forest along with SVR and Ridge Regression perform comparatively on RMSE (Root Mean Square Error) , Although Random Forest gives the best results on the test data but it is unstable and it overfits on the train data. So, It will be a wise decision to use either KNN or Ridge Regression for deployment. Model comparison table is given below.



```
   rmse                     Model
0  3.97            KNN Regression
1  3.98      Ordinary Least Square
2  3.98          Ridge Regression
3  3.99          Lasso Regression
4  4.03  Support Vector Regression
5  4.01             Decision Trees
6  4.01                      GBDT
7  3.91             Random Forest
```

So, It is obvious from above model performance comparison table, both Random Forest, SVR and Ridge Regression perform comparatively on RMSE (Root Mean Square Error) and can be used for deployment.

Appendix A - Extra Figures

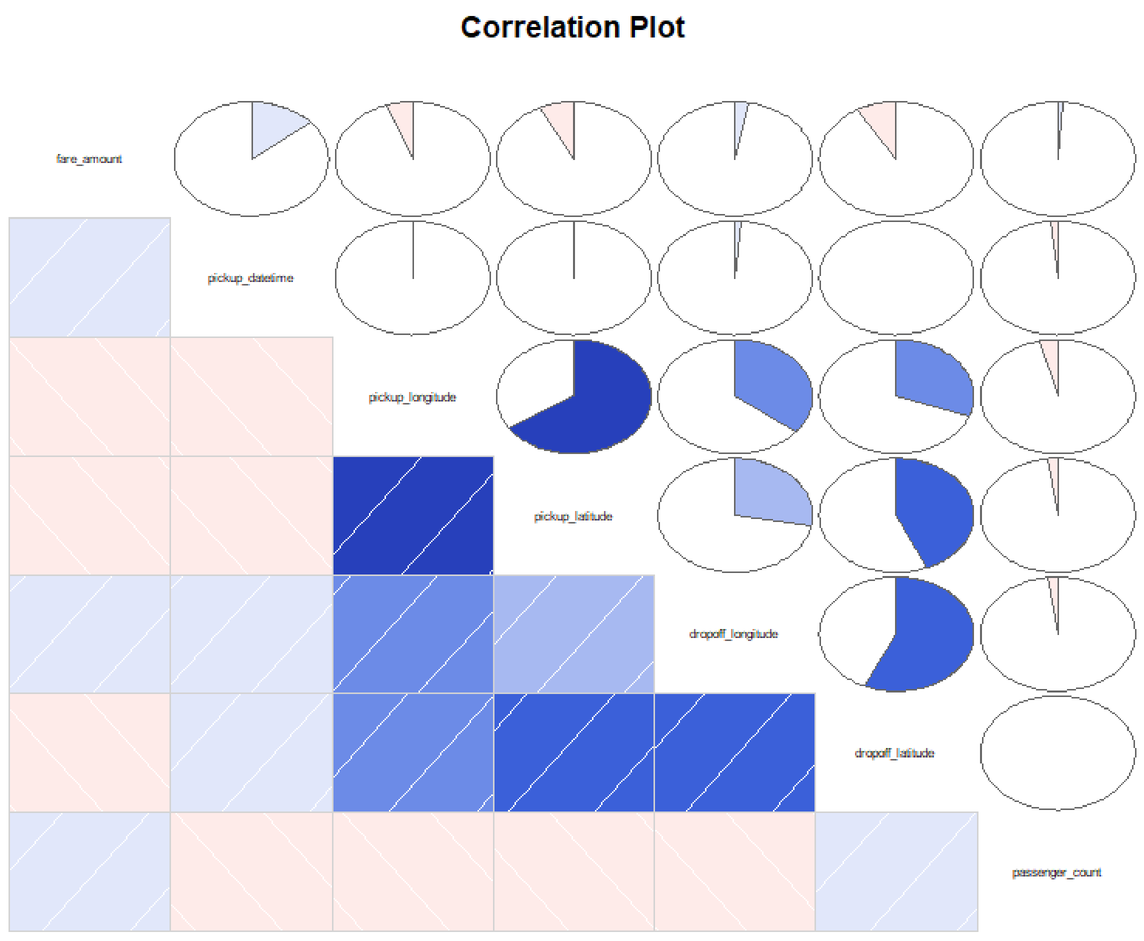


Fig : Correlation Plot in R

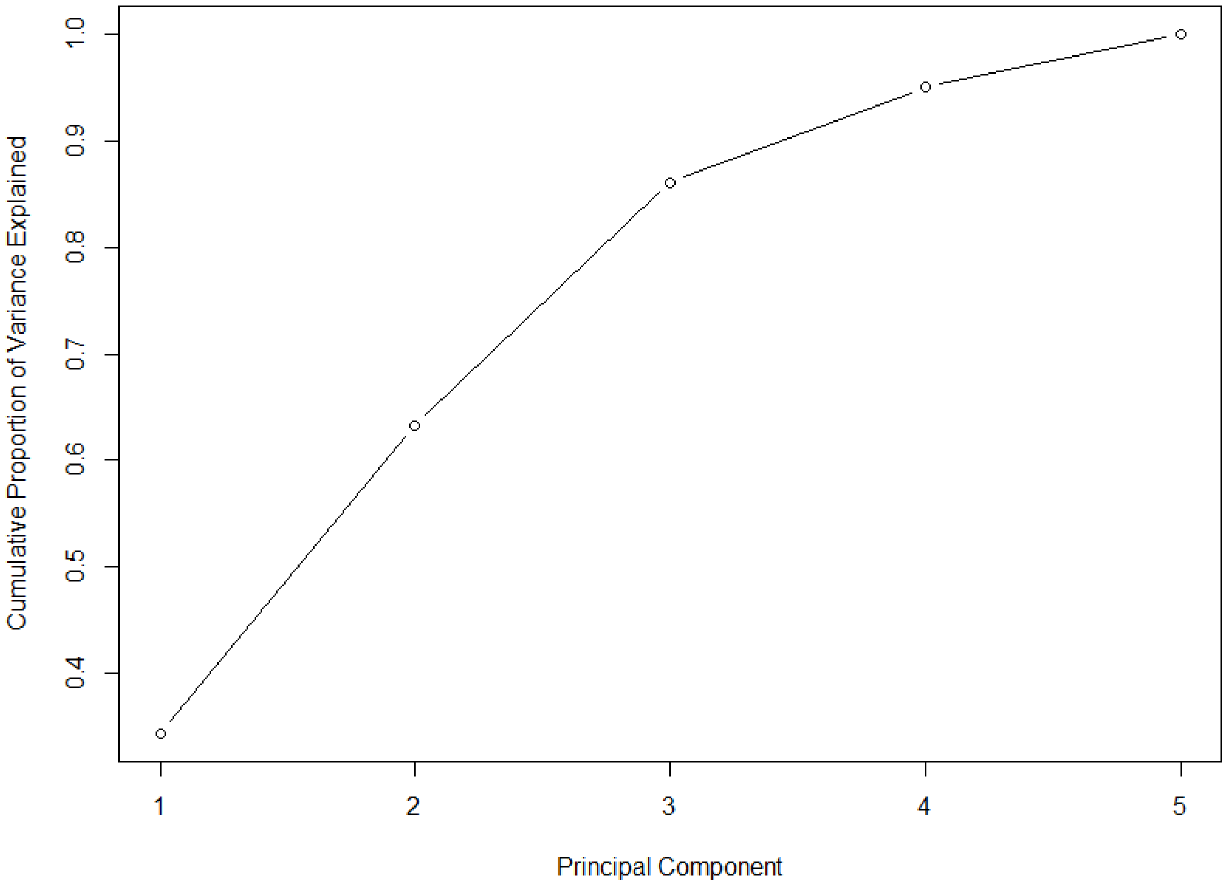
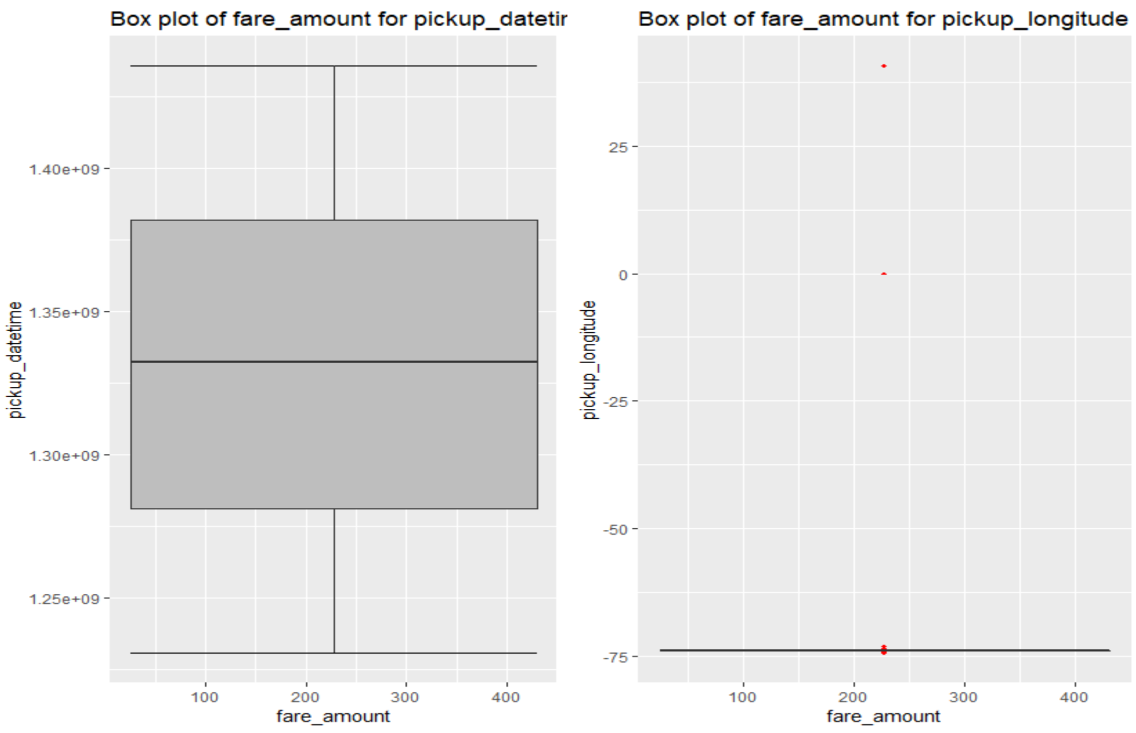
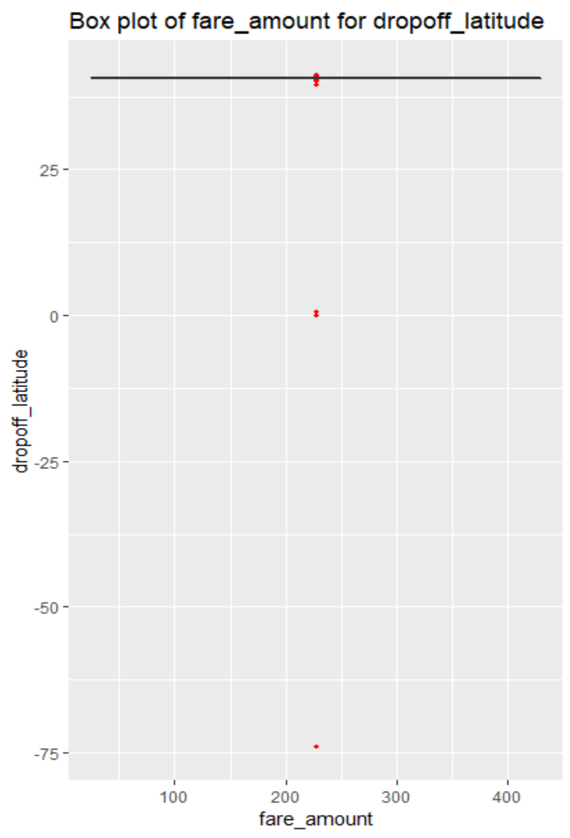
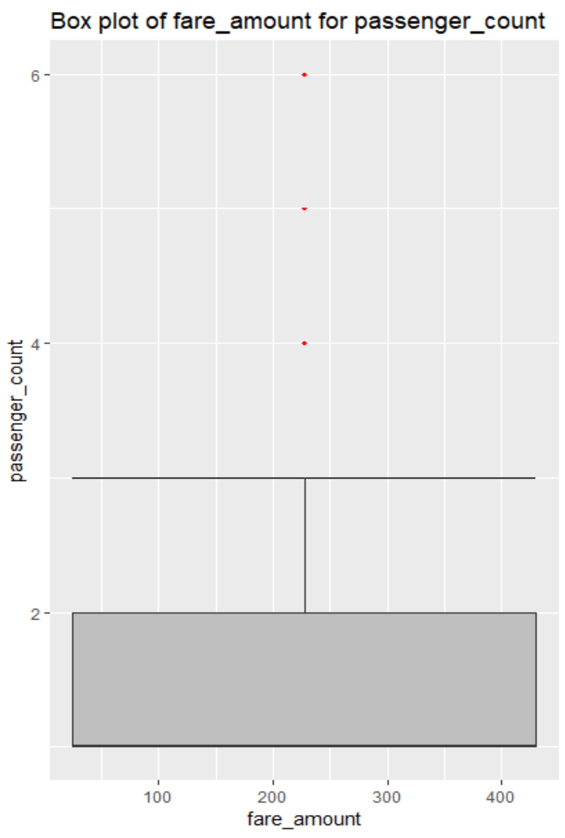


Fig : PCA Analysis in R

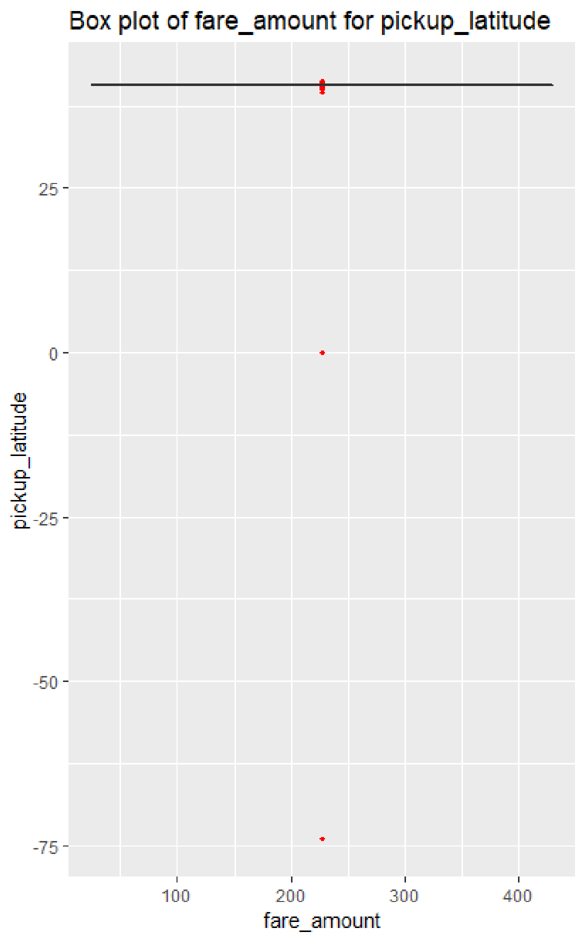




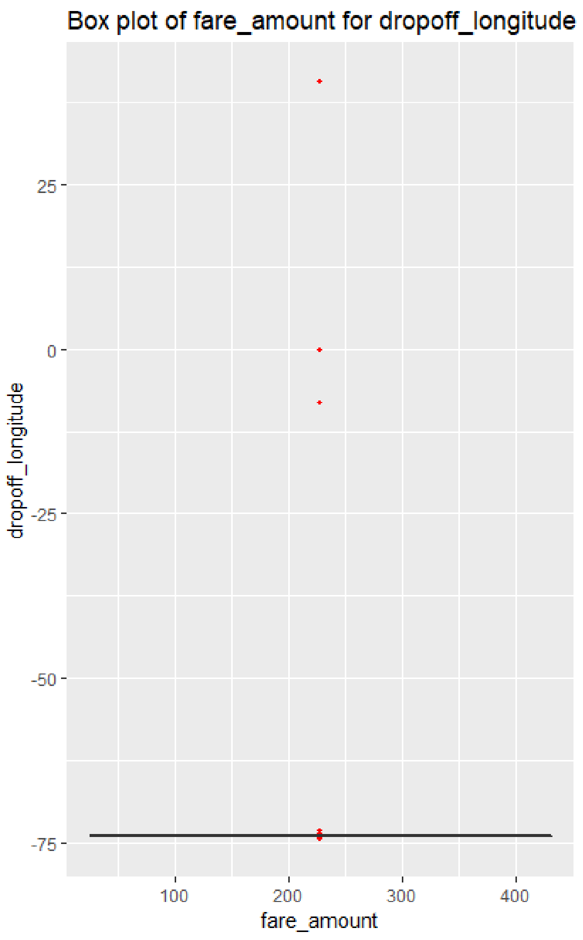
Box plot of fare_amount for dropoff_latitude
dropoff_latitude
fare_amount
25
0
-25
-50
-75
100
200
300
400



Box plot of fare_amount for passenger_count
passenger_count
fare_amount
6
4
2
100
200
300
400



Box plot of fare_amount for pickup_latitude
pickup_latitude
fare_amount
25
0
-25
-50
-75
100
200
300
400



Box plot of fare_amount for dropoff_longitude
dropoff_longitude
fare_amount
25
0
-25
-50
-75
100
200
300
400

4. REFRENCES

- Edwisor Data Science Training Path
- Couresra Machine Learning, Stanford University.
- Udacity's Data Visualization
- Statistics from DataScienceCentral blog and TowardsDataScience blogs.
- Solved errors by searching same errors on Stackoverflow.