Project Report
Cab Fare Prediction

Submitted By -
Nitisha Yadav

Contents

A problem statement in Data Science can be solved by following the below steps:

1. Define Problem Statement/ Business Requirement
2. Data Collection
3. Data Pre-processing
4. Data Exploration & Analysis
5. Data Modelling
6. Model Evaluation

1. **Problem Statement:** To build a model that will predict the fare amount for the cab ride in the city from the historical data collected from the pilot project. The fare amount will be predicted based on the fare of the pilot project which is running.

2. **Data Collection:** To get the best results it is very important to understand the data. Here data consists of 16048 observations and 7 variables. The different variables of the data are:
   - pickup_datetime - timestamp value indicating when the cab ride started.
   - pickup_longitude - float value indicating longitude coordinate of where the cab ride started.
   - ·pickup_latitude - float value indicating latitude coordinate of where the cab ride started.
   - dropoff_longitude - float value indicating longitude coordinate of where the cab ride ended.
   - ·dropoff_latitude - float value indicating latitude coordinate of where the cab ride ended.
   - passenger_count - an integer indicating the number of passengers in the cab ride
   - Fare_amount - a float value indicating the fare of the ride.

The data consists of 6 independent variables and 1 dependent variable

Independent Variables:

1. Pickup_datetime
2. Pickup_longitude
3. Pickup_latitude
4. Dropoff_longitude
5. Dropoff_latitude
6. Passenger_count

Dependent Variable:

1. Fare_amount

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
1	4.5	2009-06-15 17:26:21 UTC	-73.84431	40.72132	-73.84161	40.71228	1
2	16.9	2010-01-05 16:52:16 UTC	-74.01605	40.71130	-73.97927	40.78200	1
3	5.7	2011-08-18 00:35:00 UTC	-73.98274	40.76127	-73.99124	40.75056	2
4	7.7	2012-04-21 04:30:42 UTC	-73.98713	40.73314	-73.99157	40.75809	1
5	5.3	2010-03-09 07:51:00 UTC	-73.96810	40.76801	-73.95665	40.78376	1
6	12.1	2011-01-06 09:50:45 UTC	-74.00096	40.73163	-73.97289	40.75823	1
7	7.5	2012-11-20 20:35:00 UTC	-73.98000	40.75166	-73.97380	40.76484	1
8	16.5	2012-01-04 17:22:00 UTC	-73.95130	40.77414	-73.99009	40.75105	1
9	*NA*	2012-12-03 13:10:00 UTC	-74.00646	40.72671	-73.99308	40.73163	1
10	8.9	2009-09-02 01:11:00 UTC	-73.98066	40.73387	-73.99154	40.75814	2
11	5.3	2012-04-08 07:30:50 UTC	-73.99634	40.73714	-73.98072	40.73356	1

3. **Data Pre-processing** - This stage includes removing NA values, getting rid of redundant variables and any inconsistencies in the data. It is a data mining process that involves transformation of incomplete, inconsistent data which will be useful for building our model well. The data which we collect contains errors so we also remove them in this process.

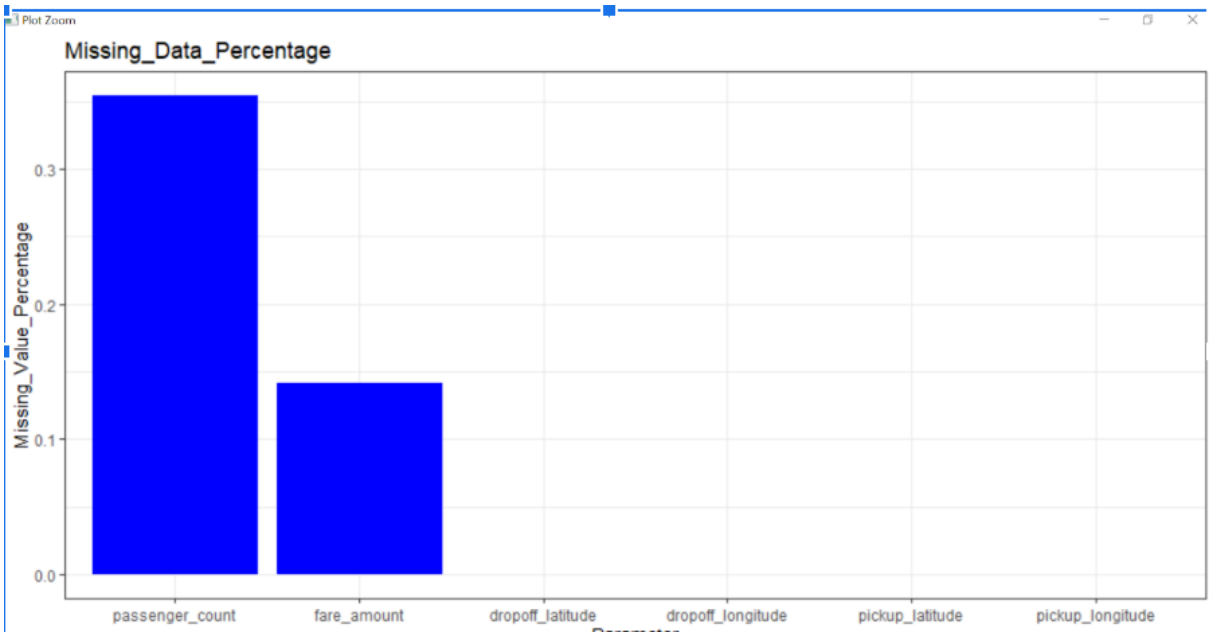
   a. **Removing Observations and Variables -** In this process the Observations and Variables are removed which do not contribute much in the model development.

   - Pickup_datetime variable is removed from the dataset as the data contains only pickup data time and not the dropoff time and date . So, it does not contribute much information in fare prediction.
   - Removing observations in which the fare amount is less than one as fare cannot be negative. In the given data there are some observations which have negative fare amount and fare amount cannot be negative.
   - Removing observations in which the passenger count is more than 6 and less than one. Data contains observation in which the passenger count is more than 10 which cannot be possible. A rental car has a capacity of maximum 6 passengers .So, removing that observation as it creates inconsistency in our dataset.
   - Removing observations in which the pickup latitude and longitude is equal to drop off latitude and longitude. As this observation carry no information because the cab had not travelled anywhere as both coordinates are same
   - Removing observations in which the pickup latitude is greater than 90 as latitude values ranges from -90 to 90.

   b. **Converting Data Types -** Converting the data type of variables which are not appropriate.

   - Fare-amount - Converting the type of fare_amount variable from factor to numeric and float in R and Python respectively.
   - Passenger count – Converting the type of passenger_count variable to integer as passenger count be a decimal number.

c. **Missing value analysis** - Missing value means there are some observation in the data which are incomplete which affects the accuracy of the model.It occurs in the data due to manual errors , wrong input, incomplete observations etc.There were various missing values in the dataset which are converted to NA and then its value is imputed using mean ,mode and knn imputations method.



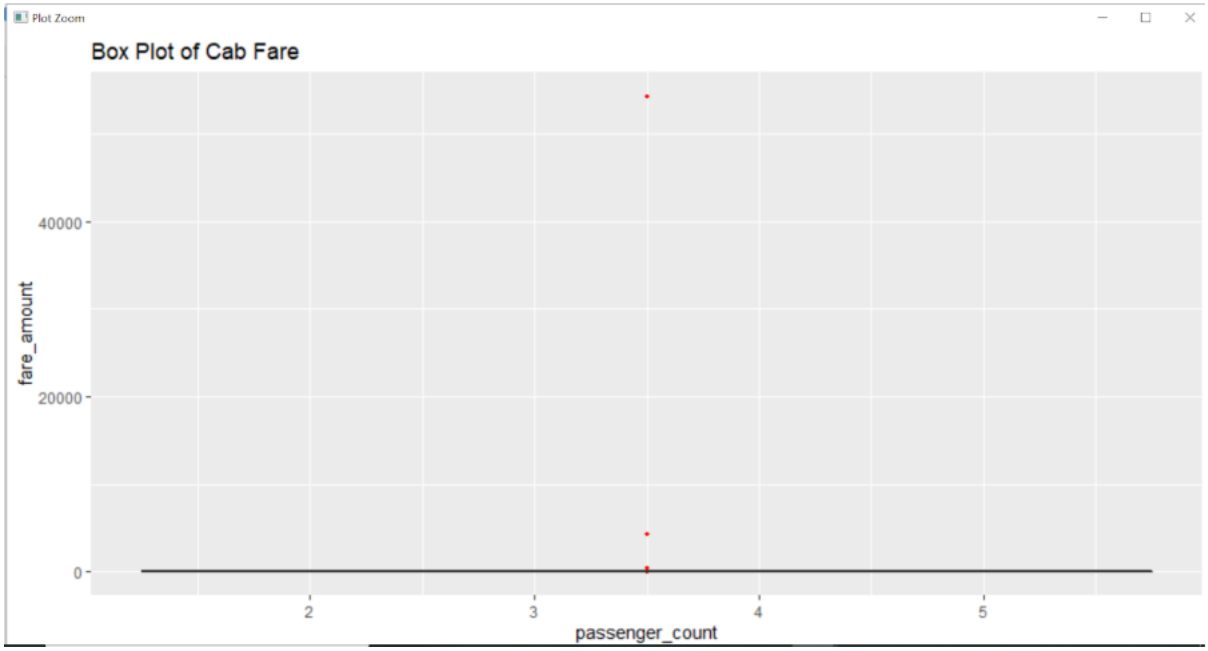
In this dataset after checking among mean ,median and knn imputation method the method which gives values original value is knn imputation.So in this the missing value is imputed using knn imputation .
Dataset after imputing missing values

fare_amount	0
pickup_longitude	0
pickup_latitude	0
dropoff_longitude	0
dropoff_latitude	0
passenger_count	0

d. **Outlier Analysis -** Outlier are the observations which are inconsistent with the rest of the dataset.It the the observations that deviate away from the other observations. It is due to poor data quality or contamination,low

quality measurements,malfunctioning equipment,manual error.Outlier causes an error in predicting the target variable.

- Fare_amount - In this dataset there are fare amounts which are too high which are out of range and that values are converted to NA and imputed using Knn method.



e. **Feature Selection** :- The variables present in our dataset is not enough to predict the fare amount of the cab.So, using the variables present in the dataset a variable is generated which will be used for prediction .The new generated variable is distance.

The fare is predicted using distance variable .This method is also known as Feature Engineering.

The function which is used to derive new variable is Haversine Function.Using this function distance is calculated from the given pickup and dropoff latitudes and longitudes. It gives the output in meters.

Haversine Function Used In R

```
library(geosphere)
train_cab$distance=distHaversine(cbind(train_cab$pickup_longitude,train_cab$pickup_latitude),cbind(train_cab$dropoff_longitude,train_cab$dropoff_latitude))
```

Haversine Function Used In Python

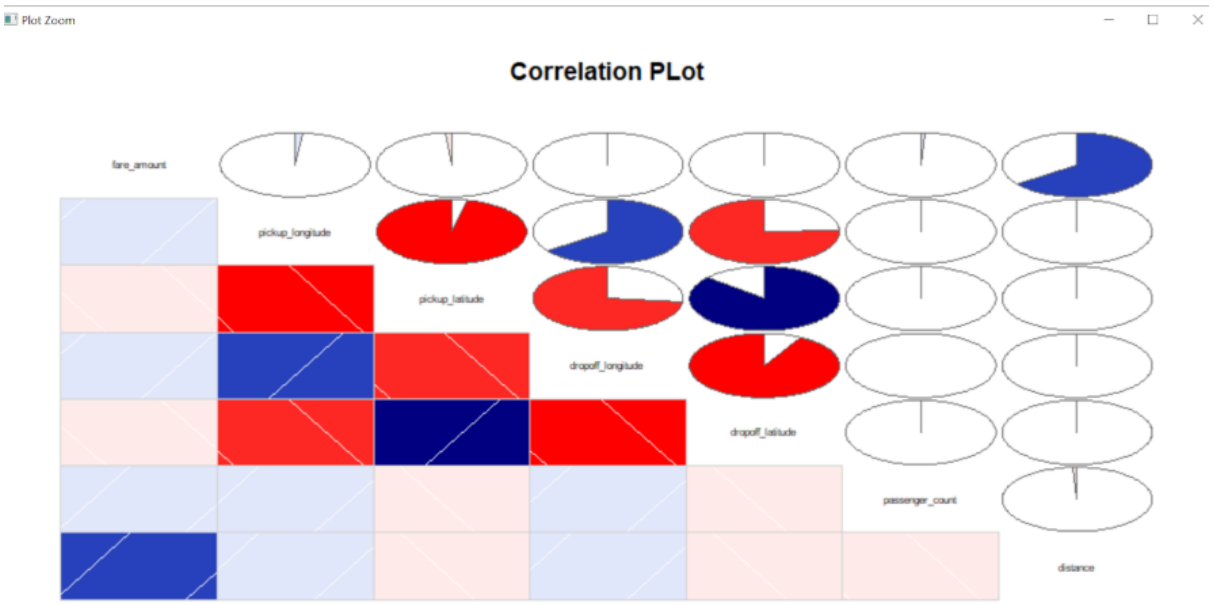
```
def haversine(lat1, lon1, lat2, lon2, to_radians=True, earth_radius=6371):

    if to_radians:
        lat1, lon1, lat2, lon2 = np.radians([lat1, lon1, lat2, lon2])
    d = np.sin((lat2-lat1)/2.0)**2 + \
        np.cos(lat1) * np.cos(lat2) * np.sin((lon2-lon1)/2.0)**2
    return earth_radius * 2 * np.arcsin(np.sqrt(d))
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	distance
1	4.500000	-73.84431	40.72132	-73.84161	40.71228	1	1031.9186
2	16.900000	-74.01605	40.71130	-73.97927	40.78200	1	5382.1559
3	5.700000	-73.98274	40.76127	-73.99124	40.75056	2	1391.0818
4	7.700000	-73.98713	40.73314	-73.99157	40.75809	1	2802.4061
5	5.300000	-73.96810	40.76801	-73.95665	40.78376	1	2001.3963
6	12.100000	-74.00096	40.73163	-73.97289	40.75823	1	3791.4817
7	7.500000	-73.98000	40.75166	-73.97380	40.76484	1	1557.5495
8	16.500000	-73.95130	40.77414	-73.99009	40.75105	1	4160.0994
9	5.840085	-74.00646	40.72671	-73.99308	40.73163	1	1254.6354
10	8.900000	-73.98066	40.73387	-73.99154	40.75814	2	2852.8190
11	5.300000	-73.99634	40.73714	-73.98072	40.73356	1	1376.1168
13	4.100000	-73.99160	40.74471	-73.98308	40.74468	2	718.5716
14	7.000000	-74.00536	40.72887	-74.00891	40.71091	1	2021.6460

Correlation Analysis -:

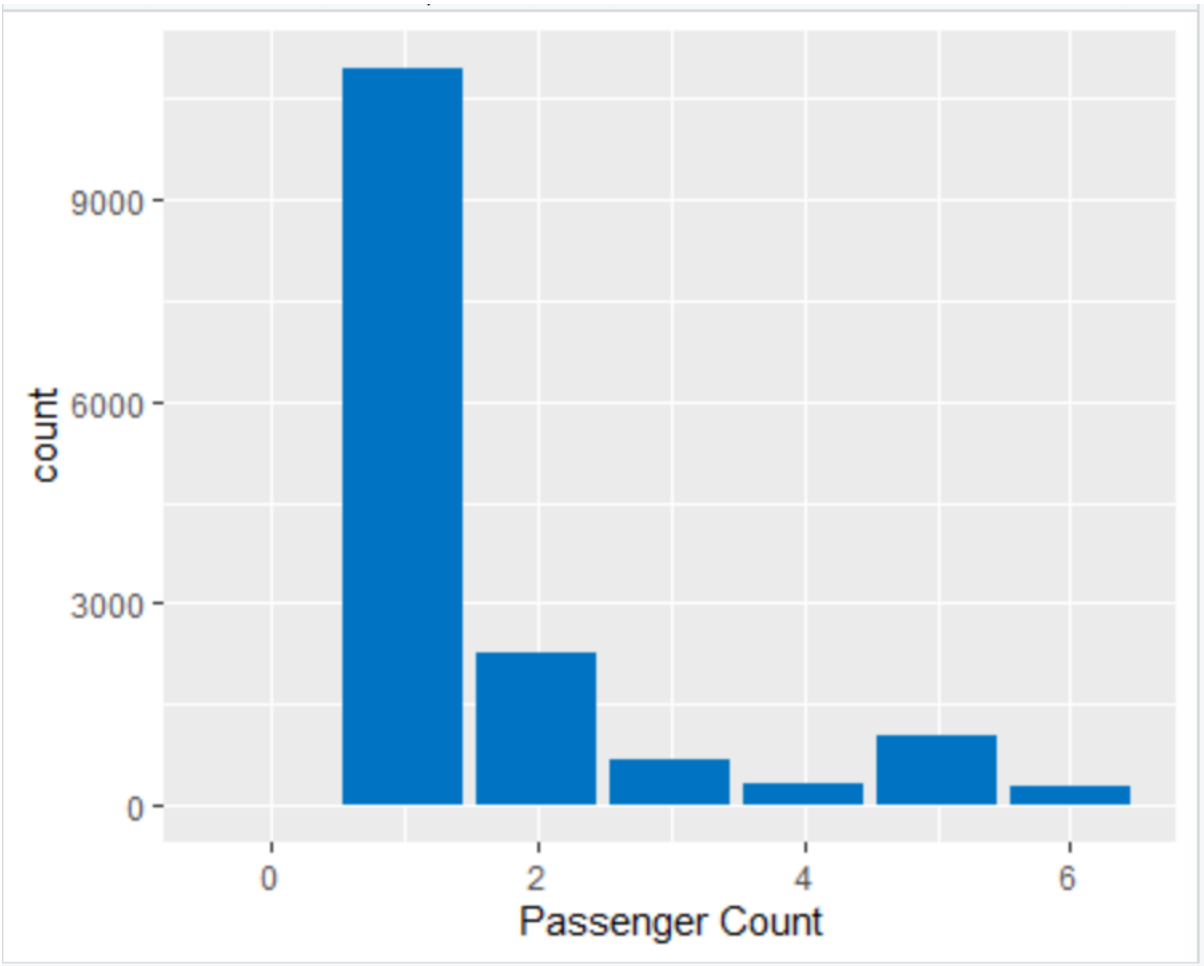
Correlation Analysis is used to define the relation between the variables.It tells the association between two continuous variables.Value of correlation analysis ranges from -1 to +1. It measures the direction and strength of the linear relationship between two quantitative variables.As our target variable is continuous so we will plot a correlation graph to show relation between variables.

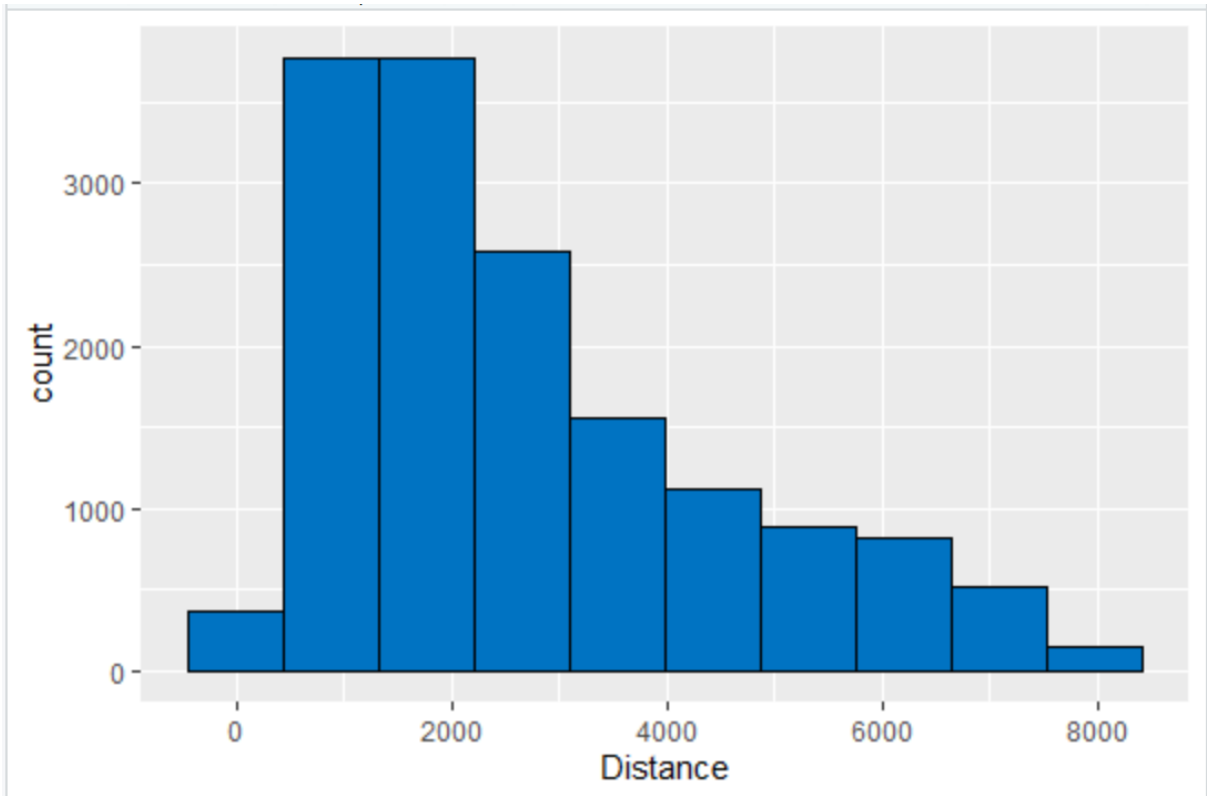


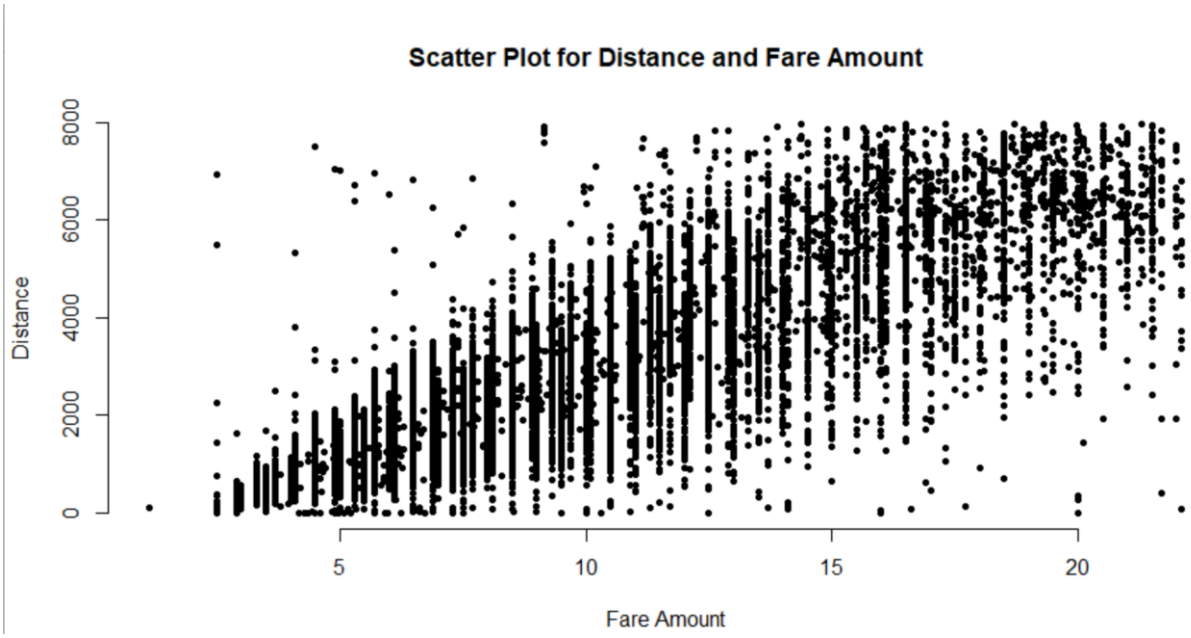
From the above visualization it is clear that most of the variables are highly correlated with each other.The dark red color shows that the variables are negatively correlated with each other and the dark blue color shows that variables are positively correlated with each other.The target variable fare amount has a positive correlation with independent variable distance.

4. Data Exploration and Analysis :-

The dataset of cabfare is pre processes using various methods and is now ready for model development. After exploring the dataset and looking at each variable it is found that the **distance** variable which is generated from pickup and drop off latitude and longitude plays a crucial role in predicting the fare amount as almost fare of any ride depends on the distance travelled.


9000
6000
3000
0
count
0
2
4
6
Passenger Count


3000
2000
1000
0
count
0
2000
4000
6000
8000
Distance



From the above scatter plot it is clear that as the distance of the ride is increasing the fare amount of the cab ride is also increasing. So the target variable fare amount is highly dependent on the independent variable distance.

5. Model Development :-

After preparing the data such as cleaning and analysis the next step is to develop a model on the processed data. As the target variable is continuous that is why it comes under Regression problem statement. The algorithms which is used for Regression problem solving are Decision Trees algorithm, Random Forest ,Linear Regression and Knn Imputation.

5.1. Decision Trees Algorithm -

Decision tree is a predictive model based on a branching series of boolean tests.

Decision Tree in R:

```
fit=rpart(fare_amount ~.,data = trained_data,method = "anova")
predictions=predict(fit,validate_data[,-1])
```

```
> fit
n= 12421

node), split, n, deviance, yval
      * denotes terminal node

1) root 12421 246161.200  9.435706
  2) distance< 3352.779 8837  62591.570  7.337444
    4) distance< 1876.302 5307  21849.800  6.100774
      8) distance< 1238.712 2959   9704.489  5.366199 *
      9) distance>=1238.712 2348   8536.468  7.026501 *
    5) distance>=1876.302 3530  20423.470  9.196654 *
  3) distance>=3352.779 3584  48731.470 14.609350
    6) distance< 4837.174 1667  13956.340 12.414260 *
    7) distance>=4837.174 1917  19758.120 16.518170 *
```

Decision Trees In Python -

```
#Decision Trees
fit_DT = DecisionTreeRegressor(max_depth=2).fit(train_data.iloc[:, 1:7], train_data.iloc[:,0])
```

```
fit_DT
```

```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

```
#Applying Model on Test Data
predictions_DT = fit_DT.predict(test_data.iloc[:,1:7])
```

```
predictions_DT
```

```
array([ 5.51872746,  5.51872746, 12.69472332, ..., 12.69472332,
       12.69472332,  7.69041765])
```

5.2. Linear Regression -

Linear Regression models are used to show or predict the relationship between two variables or factors.The factor that is being predicted (the factor that the equation *solves for*) is called the dependent variable. The factors that are used to predict the value of the dependent variable are called the independent variables.

Linear Regression in R -

```
lm_model=lm(fare_amount ~.,data = trained_data)
summary(lm_model)
predictions_lr=predict(lm_model,validate_data[,-1])
```

```
Call:
lm(formula = fare_amount ~ ., data = trained_data)

Residuals:
     Min       1Q   Median       3Q      Max
-15.8633  -1.3513  -0.4087   0.9718  17.0687

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        5.409e+00  9.851e-01   5.491 4.08e-08 ***
pickup_longitude  -3.137e-03  3.634e-02  -0.086 0.931209
pickup_latitude   -5.240e-02  5.814e-02  -0.901 0.367422
dropoff_longitude  2.347e-02  1.870e-02   1.255 0.209412
dropoff_latitude   4.694e-02  3.835e-02   1.224 0.220996
passenger_count    5.627e-02  1.641e-02   3.429 0.000607 ***
distance           2.111e-03  1.156e-05 182.647  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.318 on 12414 degrees of freedom
Multiple R-squared:  0.729,     Adjusted R-squared:  0.7289
F-statistic:  5565 on 6 and 12414 DF,  p-value: < 2.2e-16
```

Linear Regression in Python -

```
#Linear Regression
import statsmodels.api as sm
```

```
model = sm.OLS(train_data.iloc[:, 0], train_data.iloc[:, 1:7]).fit()
```

```
model.summary()
```

```
predictions_LR = model.predict(test_data.iloc[:,1:7])
```

Linear Model Summary

OLS Regression Results

Dep. Variable:	fare_amount	**R-squared:**	0.837
Model:	OLS	**Adj. R-squared:**	0.837
Method:	Least Squares	**F-statistic:**	1.054e+04
Date:	Tue, 25 Feb 2020	**Prob (F-statistic):**	0.00
Time:	16:41:38	**Log-Likelihood:**	-34534.
No. Observations:	12360	**AIC:**	6.908e+04
Df Residuals:	12354	**BIC:**	6.912e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
pickup_longitude	-0.2977	0.043	-6.889	0.000	-0.382	-0.213
pickup_latitude	-0.4567	0.078	-5.833	0.000	-0.610	-0.303
dropoff_longitude	0.0049	0.039	0.127	0.899	-0.071	0.081
dropoff_latitude	0.1433	0.070	2.034	0.042	0.005	0.281
passenger_count	0.0337	0.028	1.190	0.234	-0.022	0.089
distance	0.0007	0.000	5.700	0.000	0.000	0.001

Omnibus:	1822.124	**Durbin-Watson:**	2.011
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2761.526
Skew:	1.072	**Prob(JB):**	0.00
Kurtosis:	3.876	**Cond. No.**	905.

5.3. Random Forest Model -

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

Random Forest In R

```
library(randomForest)
RF_model = randomForest(fare_amount ~.  , trained_data, importance = TRUE, ntree=100)
RF_Predictions = predict(RF_model, validate_data[,-1])
```

```
> importance(RF_model, type = 1)
                   %IncMSE
pickup_longitude   20.30734
pickup_latitude    18.48361
dropoff_longitude  21.99781
dropoff_latitude   17.87114
passenger_count    11.70050
distance          112.63345
```

Random Forest In Python

```
#Random Forest
from sklearn.ensemble import RandomForestRegressor
```

```
RF_model = RandomForestRegressor(n_estimators = 10).fit(train_data.iloc[:, 1:7], train_data.iloc[:,0])
```

```
RF_Predictions = RF_model.predict(test_data.iloc[:, 1:7])
```

5.4. KNN Imputation -

KNN stands for K- nearest neighbour.KNN is simple algorithm that stores all available cases and classifies new cases based on a similarity measure.
It is a supervised lazy learning algorithm and can be used for both classification and regression.

KNN Imputation in R

```
library(class)
KNN_Predictions = knn(trained_data[,2:7], validate_data[, 2:7], trained_data$fare_amount, k = 1)
```

KNN Imputation in Python

```
#KNN Imputation
from sklearn.neighbors import KNeighborsRegressor

KNN_model = KNeighborsRegressor(n_neighbors = 1).fit(train_data.iloc[: , 1:7], train_data.iloc[:, 0])

KNN_Predictions = KNN_model.predict(test_data.iloc[: , 1:7])

MAPE(test_data.iloc[:,0], KNN_Predictions)
```

6. **Model Evaluation and Deployment :-**

Model is evaluated using the error metric MAPE, which means absolute percentage error. In this error metric an error rate and accuracy of the model is calculated based on which model is selected for deployment.

MAPE Function-

```
mape=function(y,yhat){
 mean(abs((y-yhat)/y*100))
  }
```

Accuracy is a better measure to compare models to select the best model among different models trained.

Accuracy can be calculated as 1- MAPE

MAPE calculated in R

Model	Error Rate	Accuracy
Decision Tree	20.61	79.39
Linear Regression	19.28	80.72
Random Forest	17.82	82.18
KNN Imputation	25.18	74.82

MAPE calculated in Python

-

Model	Error Rate	Accuracy
Decision Tree	23.66	76.34
Linear Regression	38.57	61.43

Random Forest	20.53	79.47
KNN Imputation	25.20	74.80

According to the predictions and analysis done on different models and also considering the error metric for comparison the best model for prediction among Decision Trees , Linear Regression, Random Forest and KNN Imputation is **Random Forest Model.**

So the model which fits best to our dataset to predict the fare amount more accurately is the **Random Forest Model** .

Importing the test data now and applying Random Forest Model on it to predict the fare amount.Before applying model some preprocessing is also done on the model .Variable Distance is generated ,Pickup_ date_ time variable is removed from the test dataset and the target variable fare amount is also generated in the given test data.

Then the final prediction is written back in csv file.

```
test_data=read.csv("test.csv",header = TRUE,na.strings = c("","  ","NA"))

test_data=test_data[- which((test_data$pickup_longitude
==  test_data$dropoff_longitude) &         (test_data$pickup_latitude ==
test_data$dropoff_latitude)),]

test_data=test_data[-1]

test_data$distance=distHaversine(cbind(test_data$pickup_longitude,test_data$picku
p_latitude),cbind(test_data$dropoff_longitude,test_data$dropoff_latitude))

test_data$fare_amount=0
```

Applying Random Forest Model On Test Data

```
RF_model1 = randomForest(fare_amount ~.  , trained_data, importance =
TRUE,   ntree=100)

test_data$fare_amount = predict(RF_model1, test_data[,-7])
```

Writing back to CSV.

```
write.csv(test_data,"Predicted Cab Fare Amount.csv",row.names = F)
```