Project Report On Cab Fare Prediction

DECEMBER 28

Authored by: Himanshu behl

Logo Name

INDEX

Chapter 1

Introduction

In this report we discuss about the cab fare, in the developing world there is necessary to know about the cab rental rate and transparency about the cab fare. The ease of using the services and flexibility gives their customer a great experience with competitive prices.

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 Data

We have two sets first is train set and other is test dataset. Both of the data set have different-different data points. firstly, we evaluate the both the data set and upload in the a given platform like Jupyter notebook, R studio, visual studio code and etc.

Size of train Dataset Provided: - 16067 rows, 7 Columns (including dependent variable) Missing Values: - Yes Outliers Presented: - Yes Below mentioned is a list of all the variable names with their meanings

Size of test Dataset Provided: - 9914 rows, 6 Columns (including dependent variable) Missing Values: - Yes Outliers Presented: - Yes Below mentioned is a list of all the variable names with their meanings

i. pickup_datetime - timestamp value indicating when the cab ride started.
ii. pickup_longitude - float for longitude coordinate of where the cab ride started.
iii. pickup_latitude - float for latitude coordinate of where the cab ride started.
iv. dropoff_longitude - float for longitude coordinate of where the cab ride ended.
v. dropoff_latitude - float for latitude coordinate of where the cab ride ended.
vi. passenger_count - an integer indicating the number of passengers in the cab ride.
vii. fare_amount - fare amount (only contain in train dataset)

Chapter 2
Methodology

2.1 Pre-Processing

To predict the model, firstly we look the train dataset and manipulate the data before start the modelling. after that cleaning the data as well as visualizing the data through graph and plots. In the other words we explore the data and the data analysis. To data analysis we do following steps:

- Data exploration and Cleaning
- Missing values treatment
- Outlier Analysis
- Feature Selection and Scaling
- Visualization

2.2 Modelling

After the Pre-Processing steps, now the next step is modelling. data modelling is the process of producing a descriptive diagram of relationships between various types of information that are to be stored in a database. One of the goals of data modelling is to create the most efficient method of storing information while still providing for complete access and reporting. Data modelling is a crucial skill for every data scientist, whether you are doing research design or architecting a new data store for your company. The ability to think clearly and systematically about the key data points to be stored and retrieved, and how they should be grouped and related, is what the data modelling component of data science is all about. As per our data set following models need to be tested:

- Linear regression
- Decision Tree
- Random forest
- Gradient Boosting

2.3 Model Selection

After the pre-processing and modelling the dataset the last and final step of our methodology will be the selection of the model based on the different output and results. We have different – different parameter to train the model with our output and result. we check our modelling is suitable for our test dataset or not. Then we predict the output which we required.

Chapter 3
Pre-Processing

3.1 Data exploration and Cleaning

To start the pre-processing the data, the first step in data science is data exploring and cleaning. In the other words firstly, we deal with the missing values and outliers. to data exploration and cleaning there are following steps:

- Separate out those variables which are combined.
- negative values in fare amount are removed because negative fare is not possible or we can say these values decrease the accuracy in our prediction
- Passenger count would be max 6 in cab. So, we remove the rows having passengers counts more than 6 and less than 1.
- There are some outlier figures in the fare so we need to remove those.
- Latitudes range from -90 to 90. Longitudes range from -180 to 180. We need to remove the rows if any latitude and longitude lies beyond the ranges.

3.2 Creating some new variables from the given variables

In dataset, variable name pickup_datetime contains date and time for pickup. So we make the new variables from pickup_datetime:

- Year
- Month
- Date
- Day of Week
- Hour
- Minute

Now we find how much distance travel by the passenger from the pickup_longitude, pickup_latitude to the dropup_longitude and dropup_latitude by using the haversine formula. Form the haversine formula we find the exact distance between the two location and we easily predict the fare of a cab according to distance.

3.3 Selection of variables

Now as we know that all above variables are of now use so we will drop the redundant variables:

- pickup_datetime
- pickup_longitude
- pickup_latitude
- dropoff_longitude
- dropoff_latitude
- Minute

By adding or selecting the new variable our dataset is look like below:

	fare_amount	passenger_count	year	month	date	Day	Hour	distance
0	4.5	1.0	2009.0	6.0	15.0	0.0	17.0	1.035456
1	16.9	1.0	2010.0	1.0	5.0	1.0	16.0	8.488598
2	5.7	2.0	2011.0	8.0	18.0	3.0	0.0	1.395850
3	7.7	1.0	2012.0	4.0	21.0	5.0	4.0	2.812012
4	5.3	1.0	2010.0	3.0	9.0	1.0	7.0	2.008257

By checking the dataset data types we see our data in our proper form

```
fare_amount        float64
passenger_count      int64
year                 int64
month                int64
date                 int64
Day                  int64
Hour                 int64
distance           float64
dtype: object
```

3.4 Some more data exploration

In this report we are trying to predict the fare prices of a cab rental company. So here we have a data set of 16067 observations with 8 variables including one dependent variable.

Independent variables:
passenger_count, year, Month, Date, Day of Week, Hour, distance
Dependent variable:
fare_amount

Uniqueness in Variable:
We need to look at the unique number in the variables which help us to decide whether the variable is categorical or numeric. So, by using python script 'n unique' we tried to find out the unique values in each variable. We have also added the table below:

Variable name	Unique counts
Fare_amount	450
Passenger_count	7
Year	7
Month	12
Date	31
Day of week	7
Hour	24
Distance	15424

Dividing the variables into two categories basis their data types:

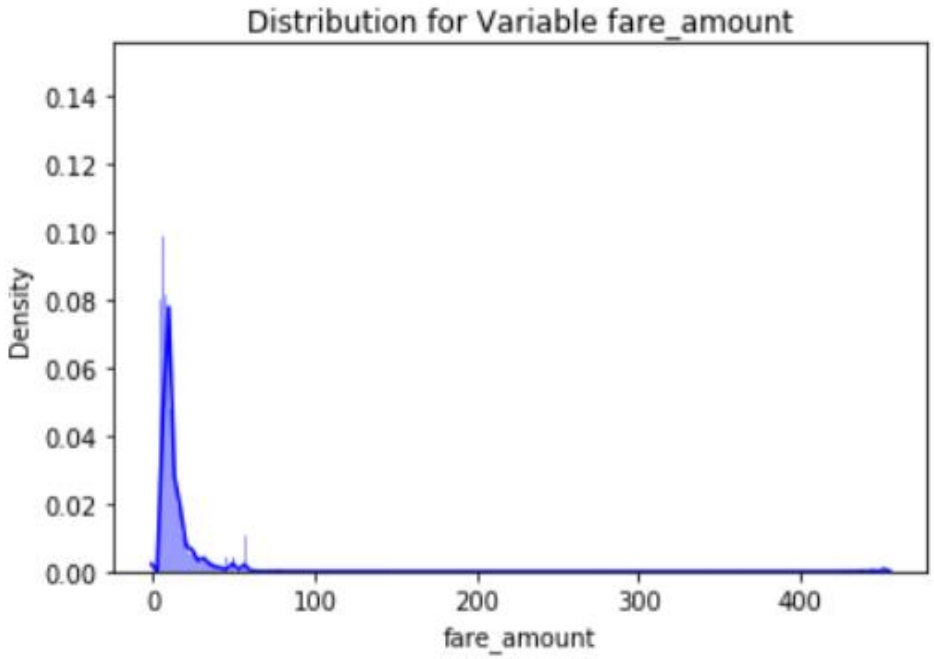
Continuous variables - 'fare_amount', 'distance'.
Categorical Variables - 'year', 'Month', 'Date', 'Day of Week', 'Hour', 'passenger_count'

3.5 Feature Scaling

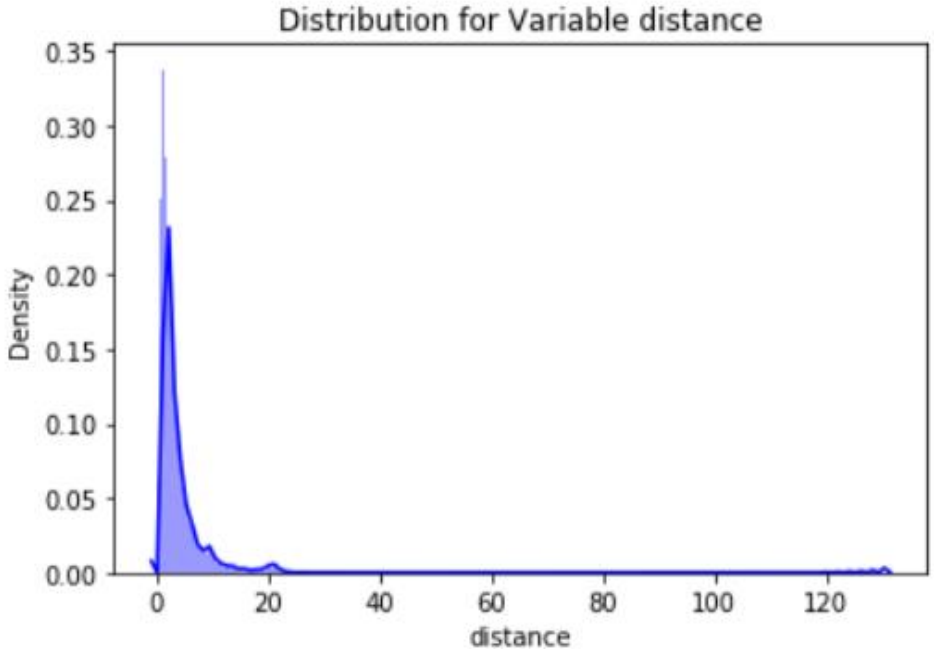
Skewness is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution. Here we tried to show the skewness of our variables and we find that our target variable absenteeism in hours

having is one sided skewed so by using log transform technique we tried to reduce the skewness of the same.

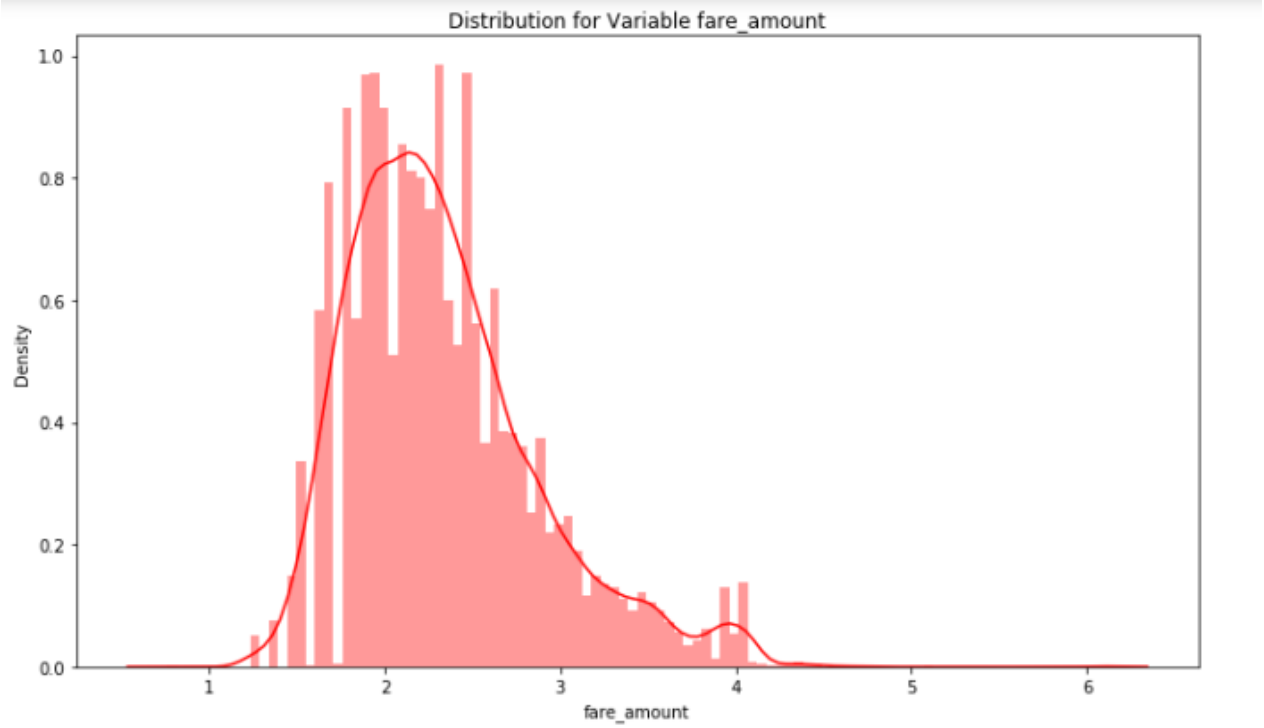
Below mentioned graphs shows the probability distribution plot to check distribution before log transformation:



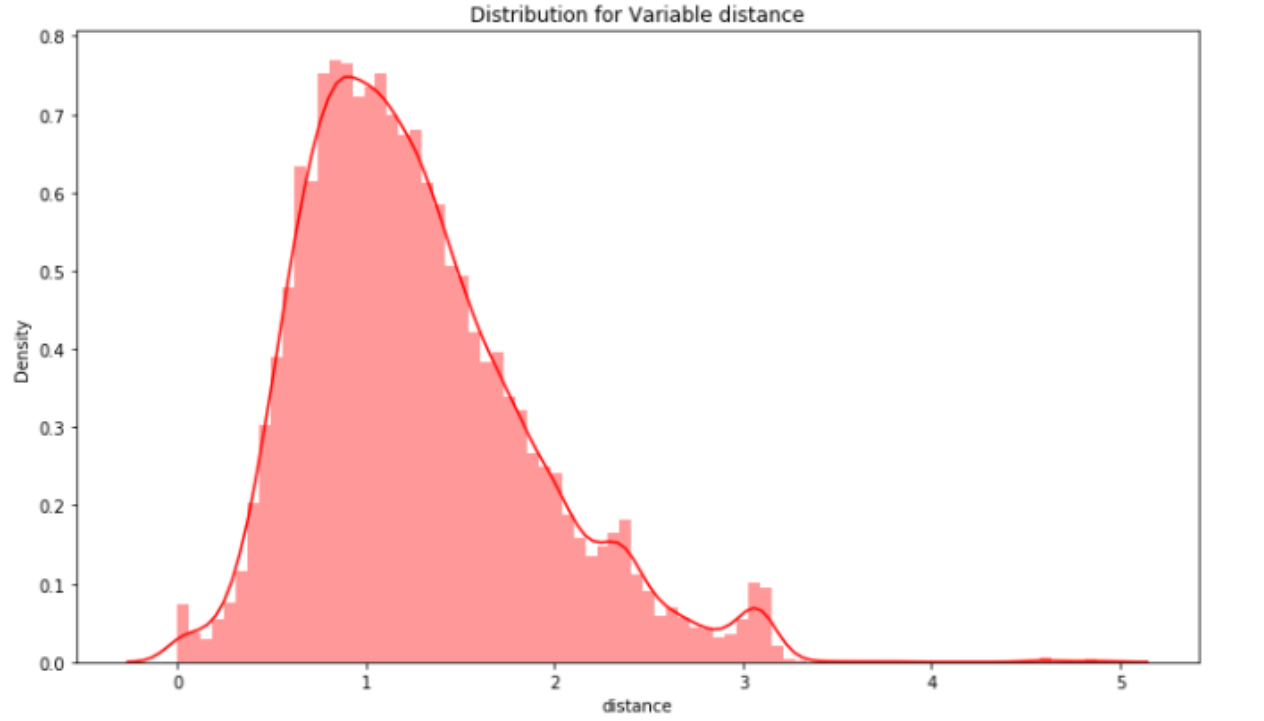
distance



Below mentioned graphs shows the probability distribution plot to check distribution after log transformation:




```
distance
```



Chapter 4
Modelling

After the pre-processing, we will use some regression models on our processed data to predict the target variable. Before running any model, we will split our data into two parts which is train and test data. Here in our case we have taken 80% of the data as train and other is test

```
#train test split for further modelling
X_train, X_test, y_train, y_test = train_test_split( df_train.iloc[:, df_train.columns != 'fare_amount'],
                    df_train.iloc[:, 0], test_size = 0.20, random_state = 1)
print(X_train.shape)
print(X_test.shape)
(12339, 7)
(3085, 7)
```

4.1 Linear regression

Linear regression is useful for finding relationship between two continuous variables. One is predictor or independent variable and other is response or dependent variable. It looks for statistical relationship but not deterministic relationship. Relationship between two variables is said to be deterministic if one variable can be accurately expressed by the other. Statistical relationship is not accurate in determining relationship between two variables. For example, relationship between height and weight.

```
# Building model on top of training dataset
fit_LR = LinearRegression().fit(X_train , y_train)
#prediction on train data
pred_train_LR = fit_LR.predict(X_train)
#prediction on test data
pred_test_LR = fit_LR.predict(X_test)
from sklearn.metrics import mean_squared_error
#calculating RMSE for test data
RMSE_test_LR = np.sqrt(mean_squared_error(y_test, pred_test_LR))
#calculating RMSE for train data
RMSE_train_LR= np.sqrt(mean_squared_error(y_train, pred_train_LR))
print("RMS Error For Training data = "+str(RMSE_train_LR))
print("RMS Error For Test data = "+str(RMSE_test_LR))
RMS Error For Training data = 0.27534581346949777 RMS Error For Test data = 0.24543871076844195
#calculate R^2 for train data
from sklearn.metrics import r2_score
r2_score(y_train, pred_train_LR)
0.7494869249991962
r2_score(y_test, pred_test_LR)
0.7826450726669177
```

4.2 Decision tree

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label.

```
fit_DT = DecisionTreeRegressor(max_depth = 2).fit(X_train,y_train)
#prediction on train data
pred_train_DT = fit_DT.predict(X_train)
#prediction on test data
pred_test_DT = fit_DT.predict(X_test)
#calculating RMSE for train data
RMSE_train_DT = np.sqrt(mean_squared_error(y_train, pred_train_DT))
#calculating RMSE for test data
RMSE_test_DT = np.sqrt(mean_squared_error(y_test, pred_test_DT))
print("RMSE For Training data = "+str(RMSE_train_DT))
print("RMSE For Test data = "+str(RMSE_test_DT))
RMSE For Training data = 0.29962109020770195 RMSE For Test data = 0.28674606171586176
# R^2 calculation for train data
r2_score(y_train, pred_train_DT)
0.7033678616157002
```

4.3 Random forest

Random forests an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the classification or regression of the individual trees. Random forest correct for decision trees' habit of overfitting to their training set

```
from sklearn.ensemble import RandomForestRegressor
fit_RF = RandomForestRegressor(n_estimators = 200).fit(X_train,y_train)
#prediction on train data
pred_train_RF = fit_RF.predict(X_train)
#prediction on test data
pred_test_RF = fit_RF.predict(X_test)
#calculating RMSE for train data
RMSE_train_RF = np.sqrt(mean_squared_error(y_train, pred_train_RF))
#calculating RMSE for test data
RMSE_test_RF = np.sqrt(mean_squared_error(y_test, pred_test_RF))
print("RMSE For Training data = "+str(RMSE_train_RF))
print("RMSE For Test data = "+str(RMSE_test_RF))
RMSE For Training data = 0.09611933292408252 RMSE For Test data = 0.23498980607960857
# calculate R^2 for train data
r2_score(y_train, pred_train_RF)
0.9694722399876309
#calculate R^2 for test data
r2_score(y_test, pred_test_RF)      0.8007577609698167
```

4.4 Hyper Parameters Tunings for optimizing the results

Model hyperparameters are set by the data scientist ahead of training and control implementation aspects of the model. The weights learned during training of a linear regression model are parameters while the number of trees in a random forest is a model hyperparameter because this is set by the data scientist. Hyperparameters can be thought of as model settings. These settings need to be tuned for each problem because the best model hyperparameters for one particular dataset will not be the best across all datasets. The process of hyperparameter tuning (also called hyperparameter optimization) means finding the combination of hyperparameter values for a machine learning model that performs the best - as measured on a validation dataset - for a problem.

Here we have used two hyper parameters tuning techniques

(i) Random Search CV

(ii) Grid Search CV

```
from sklearn.model_selection import GridSearchCV
# Grid Search CV for random Forest model
regr = RandomForestRegressor(random_state = 0)
n_estimator = list(range(11,20,1))
depth = list(range(5,15,2))
# Create the grid
grid_search = {'n_estimators': n_estimator,
'max_depth': depth}
# Grid Search Cross-Validation with 5 fold CV
gridcv_rf = GridSearchCV(regr, param_grid = grid_search, cv = 5)
gridcv_rf = gridcv_rf.fit(X_train,y_train)
view_best_params_GRF = gridcv_rf.best_params_
#Apply model on test data
predictions_GRF_test_Df = gridcv_rf.predict(df_test)
predictions_GRF_test_Df
array([2.36760025, 2.39383317, 1.6809062 , ..., 4.01224357, 3.29348722, 2.0360277 ])
df_test['Predicted_fare'] = predictions_GRF_test_Df
df_test.head()
```

	passenger_count	year	month	date	Day	Hour	distance	Predicted_fare
0	1	2015	1	27	1	13	1.204123	2.367600
1	1	2015	1	27	1	13	1.234422	2.393833
2	1	2011	10	8	5	11	0.483317	1.680906
3	1	2012	12	1	5	21	1.088548	2.209257
4	1	2012	12	1	5	21	1.858144	2.815112

Chapter 5
Conclusion

5.1 Model Evaluation and Selection

On the basis RMSE and R Squared results a good model should have least RMSE and max R Squared value. So, from above tables we can see:

(i) From the observation of all RMSE Value and R-Squared Value we concluded that.

(ii) Both the models- Gradient Boosting Default and Random Forest perform comparatively well while comparing their RMSE and R-Squared value.

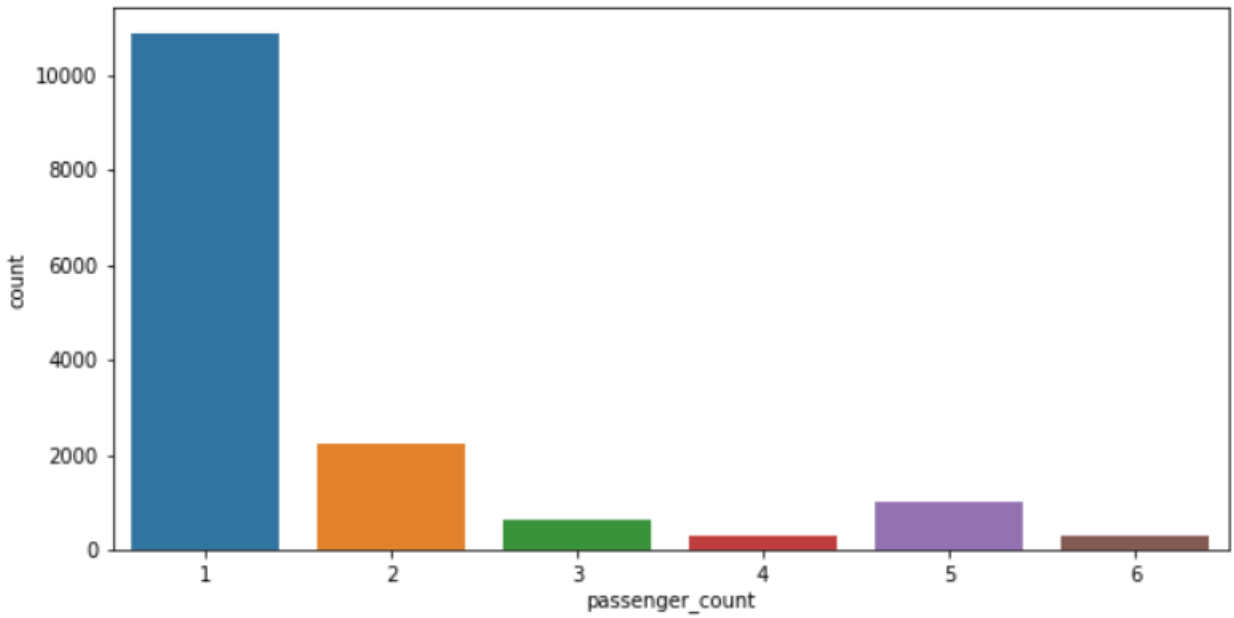
(iii) After this, I chose Random Forest CV and Grid Search CV to apply cross validation technique and see changes brought about by that.

(iv) After applying tunings Random forest model shows best results compared to gradient boosting.

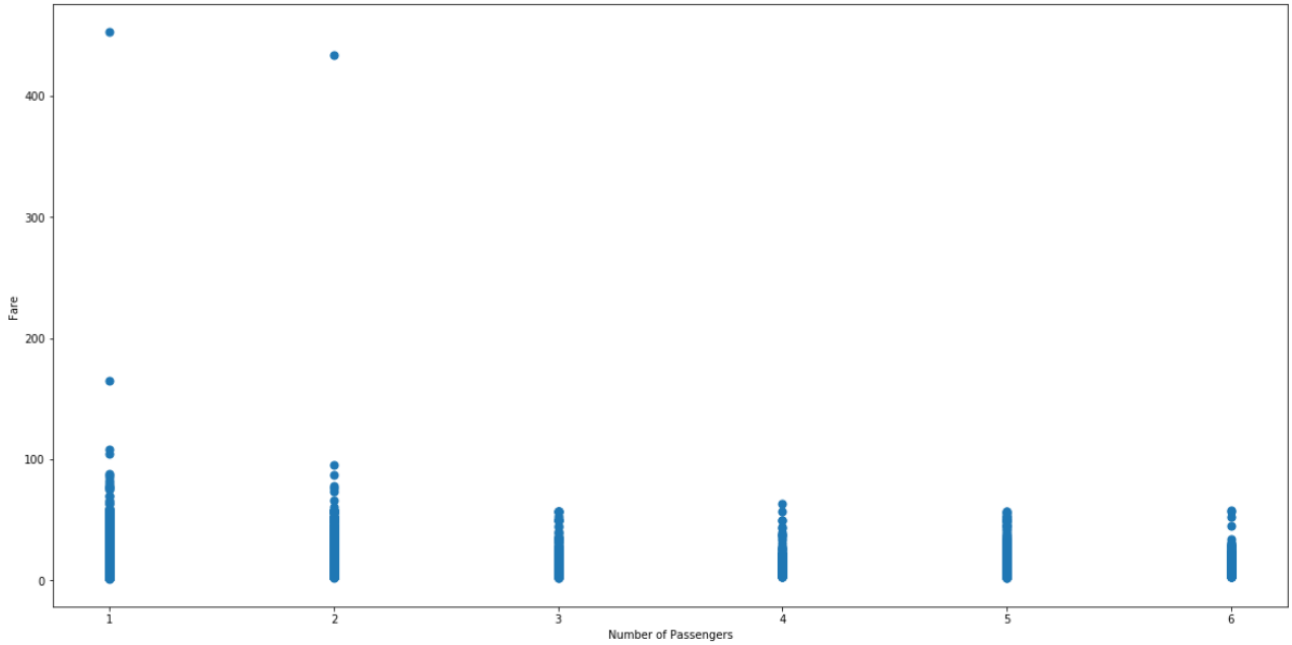
(v) So finally, we can say that Random forest model is the best method to make prediction for this project with highest explained variance of the target variables and lowest error chances with parameter tuning technique Grid Search CV.

5.2 Visualization

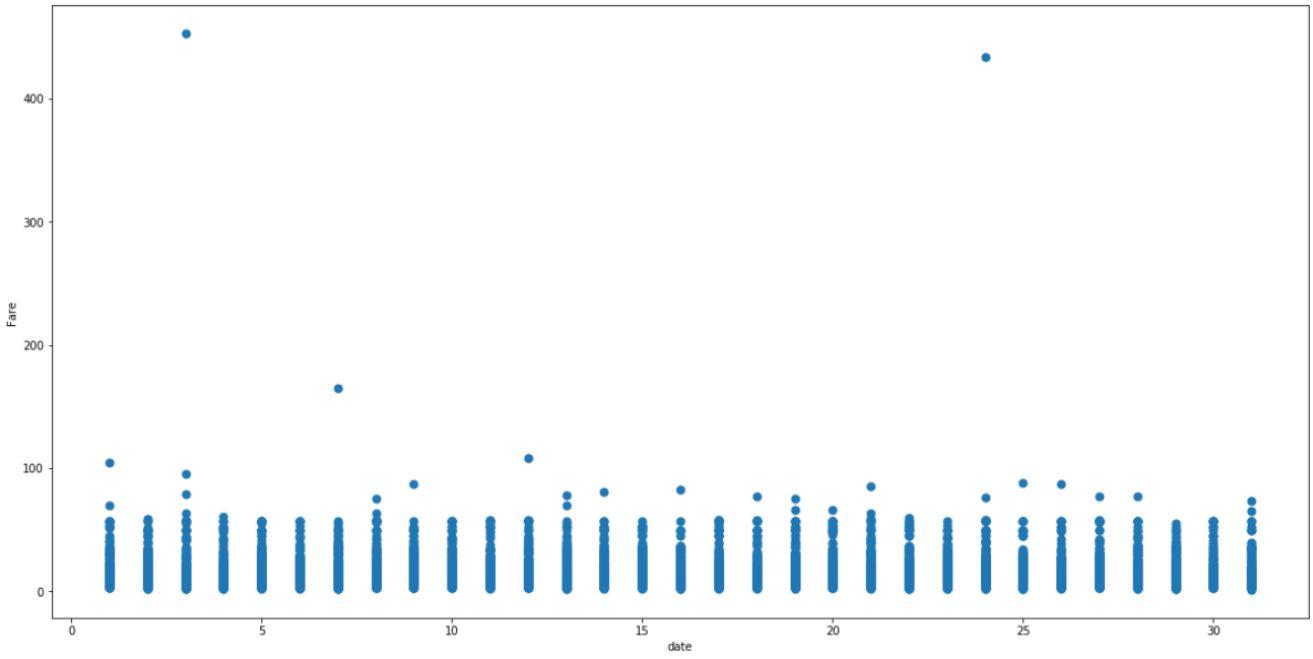
This plot the relation between the passenger count and the count and only one passenger has maximum number of counts.



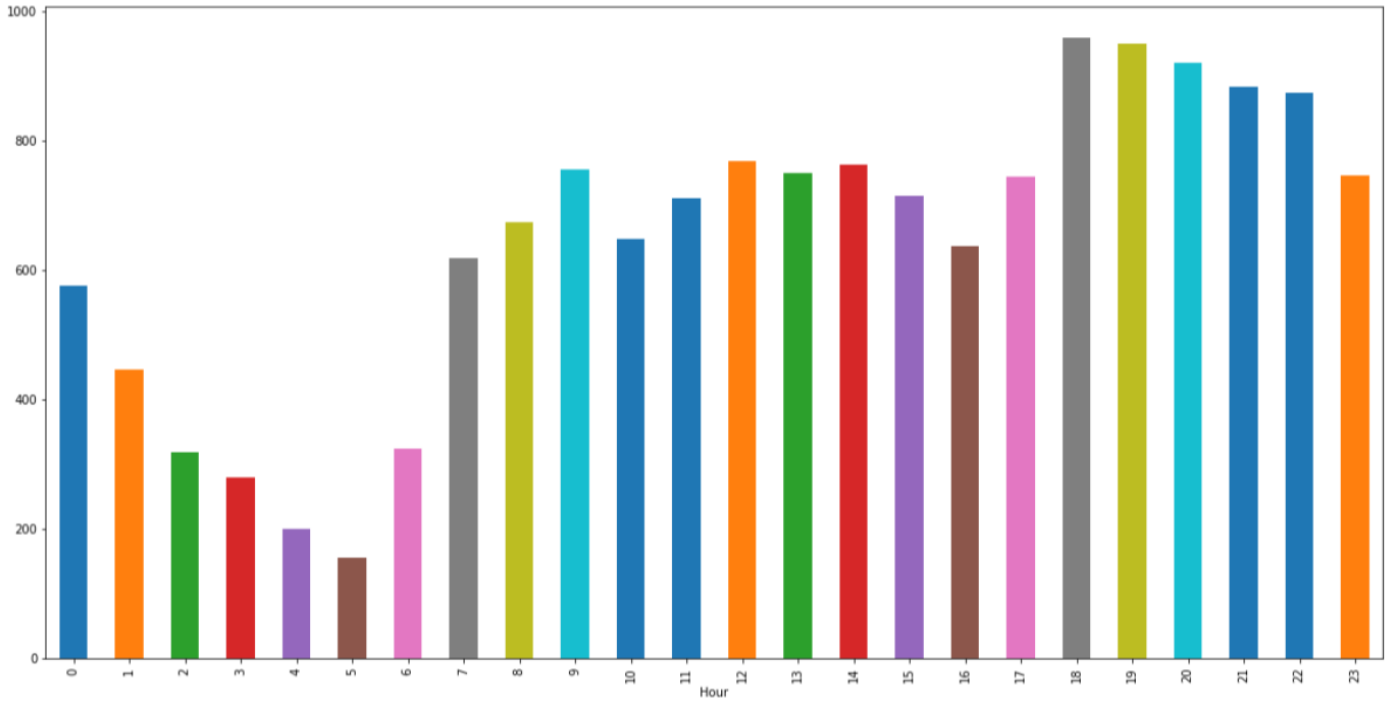
This plot shows the relation between the cab fare and the number of passengers. From the visualization almost equal car fare for number of passengers excepts outliers.



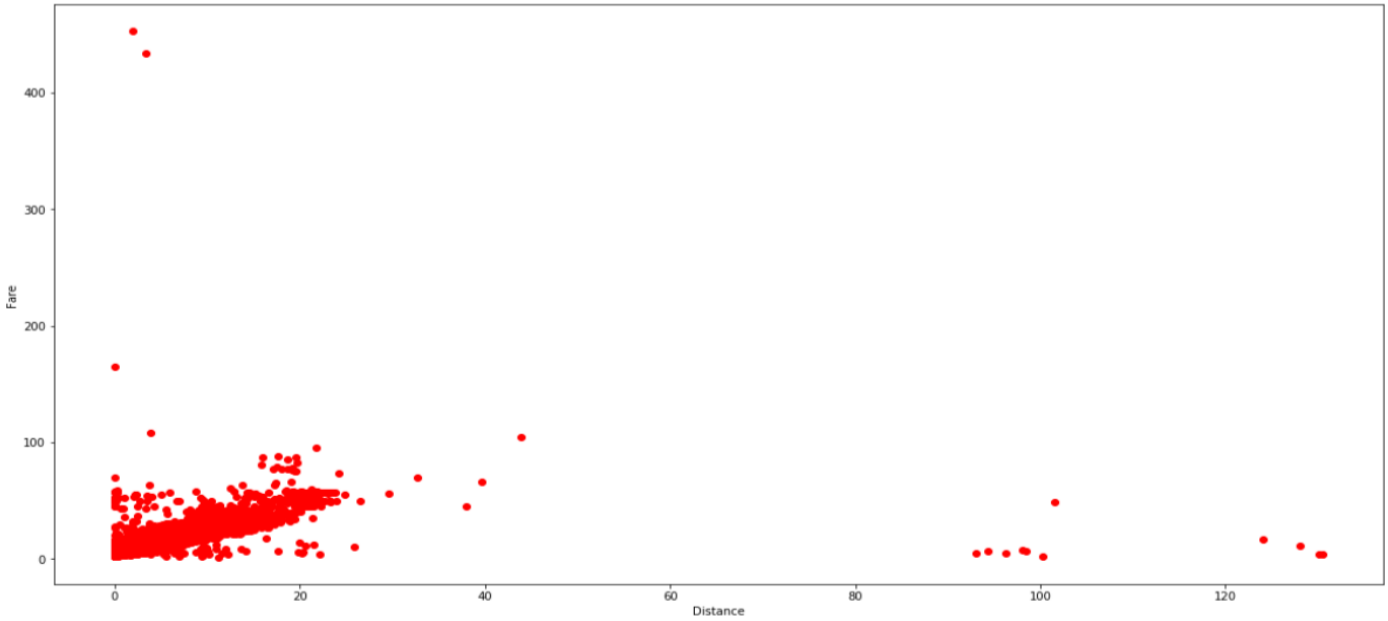
This plot shows the relation between the date and fare, from the plot we observe that there is effect on fare through out the month



This plot tells about at what time most of the passenger take the cab and from this plot we see the maximum passenger take cab in between 18:00 to 23:00 and minimum passenger take cab in between 1:00 to 6:00



From this plot we see the relation between the distance and the fare .form this plot we predict that most of the fare of cab is lies between 0 to 25



REPORT END