Project Report on Cab Fare Prediction

AUTHOR: Hamid Saifi

INDEX

Chapter 1

Introduction

Cab rental services are expanding to a large extent, with the multiplier rate. The ease of using the services and flexibility gives their customer a great experience with competitive prices. It is the need of the hour to understand the importance of technology in such aspects.

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 there is 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. And you are applying algorithms to get your accuracy and output.

1.2 Data

Understanding data is the first and important step in finding a solution to any business problem. Here in our case, our company has provided a dataset with the following features; we need to go through every variable to understand it for better functioning.
Size of 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:

Variables	Description
fare_amount	Fare amount
pickup_datetime	Cab pickup date with time
pickup_longitude	Pickup location longitude
pickup_latitude	Pickup location latitude
dropoff_longitude	Drop location longitude
dropoff_latitude	Drop location latitude
passenger_count	Number of passengers sitting in the cab

Chapter 2

Methodology

Pre-Processing

- When we are required to build a predictive model, we need to look and manipulate the data before we start modelling which includes multiple preprocessing steps such as exploring the data, cleaning the data as well as visualizing the data through graph and plots, all these steps are combined under one shed which is **Exploratory Data Analysis**, which includes the following steps:

- · Data exploration and Cleaning

- · Missing values treatment
- · Outlier Analysis
- · Feature Selection
- · Features Scaling
- · Skewness and Log transformation
- · Visualization

Modelling

- Once all the Pre-Processing steps have been done on our dataset, we will now further move to our next step, which is modelling. Modelling plays an important role to find out the useful inferences from the data. Choice of models depends upon the problem statement and dataset. As per our problem statement and dataset, we will try some models on our preprocessed data and post comparing the output results; we will select the best suitable model for our problem. As per our data set, the following models need to be tested:

- · Linear regression

- · Decision Tree

- · Random forest

- We have also used hyperparameter tunings to check the parameters on which our model runs best. We have used the following technique of hyperparameter tuning: o Random Search CV

Model Selection

- Our methodology's final step will be selecting the model based on the different output and results shown by different models. We have multiple parameters, which we will study further in our report to test whether the model is suitable for our problem statement or not.

Chapter 3

Pre-Processing

3.1 Data exploration and Cleaning (Missing Values and Outliers)

The very first step which comes with any data science project is data exploration and cleaning, which includes the following points as per this project:

a. Separating the combined variables.

b. As we know, we have some negative values in the fare amount so we need to remove those values.

c. Passenger count would be max 6 if it is an SUV vehicle, not more than that. We have to remove the rows having passengers counts more than 6 and less than 1.

d. There are some outlier figures in the fare (like top 3 values) so we need to remove those.

e. Latitudes range from -90 to 90. Longitudes range from -180 to 180. We need to remove the rows if any latitude and longitude lie beyond the ranges.

3.2 Creating some new variables from the given variables.

Here in our dataset our variable name pickup_datetime contains date and time for pickup. So we have tried to extract some important variables from pickup_datetime:

- Year
- Month
- Date
- Day
- Hour
- Minute

Also, we have tried to find out the distance using the haversine formula, which says:

The **haversine formula** determines the great-circle distance between two points on a sphere, given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles. So our new extracted variables are:

- fare_amount
- pickup_datetime
- pickup_longitude
- pickup_latitude
- dropoff_longitude
- dropoff_latitude
- passenger_count
- year
- Month
- Date Day
- Hour
- Minute 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

Now only following variables we will use for further steps:

	fare_amount	passenger_count	year	Month	Date	Day of Week	Hour	distance
0	4.5	1.0	2009.0	6.0	15.0	0.0	17.0	1.030764
1	16.9	1.0	2010.0	1.0	5.0	1.0	16.0	8.450134
2	5.7	2.0	2011.0	8.0	18.0	3.0	0.0	1.389525
3	7.7	1.0	2012.0	4.0	21.0	5.0	4.0	2.799270
4	5.3	1.0	2010.0	3.0	9.0	1.0	7.0	1.999157
5	12.1	1.0	2011.0	1.0	6.0	3.0	9.0	3.787239
6	7.5	1.0	2012.0	11.0	20.0	1.0	20.0	1.555807
8	8.9	2.0	2009.0	9.0	2.0	2.0	1.0	2.849627
9	5.3	1.0	2012.0	4.0	8.0	6.0	7.0	1.374577
10	5.5	3.0	2012.0	12.0	24.0	0.0	11.0	0.000000

VariableNames	Variable DataTypes
fare_amount	float64
passenger_count	int64
year	int64
Month	int64
Date	int64
Day	int64
Hour	int64
distance	float64

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.

3.4.1 Below are the names of Independent variables:

passenger_count, year, Month, Date, Day of Week, Hour, distance

Our Dependent variable is: **fare_amount**

3.4.2 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 'nunique' 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	7
Hour	24
distance	15424

3.4.3 Dividing the variables into two categories basis their data types:

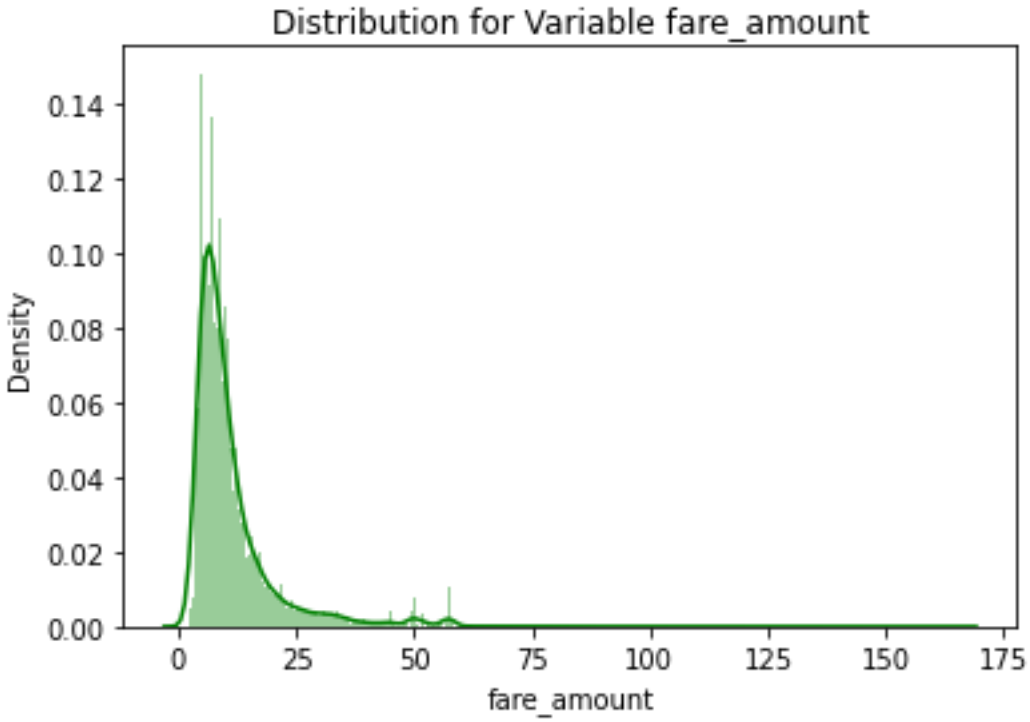
Continuous variables - 'fare_amount', 'distance'.

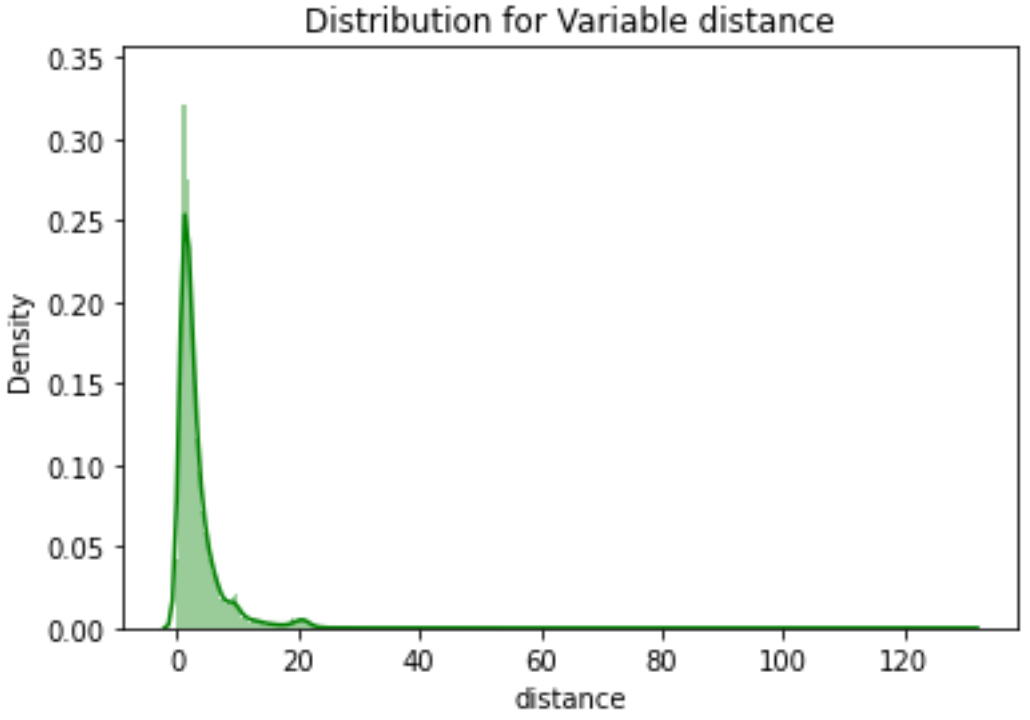
Categorical Variables - 'year', 'Month', 'Date', 'Day, 'Hour', 'passenger_count'

3.5 Feature Scaling

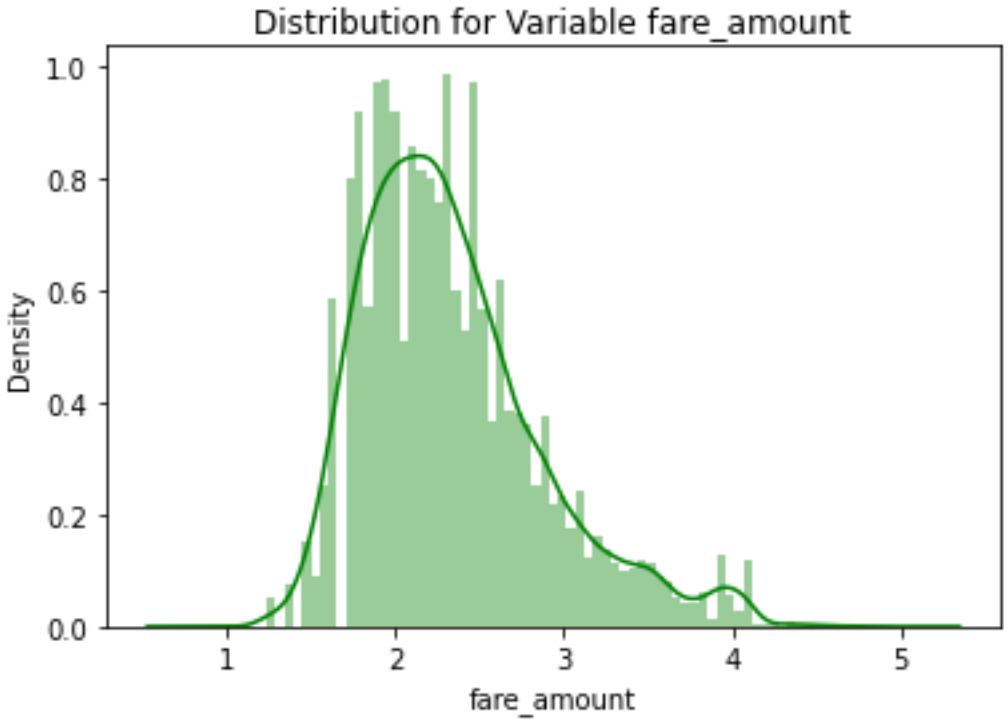
Skewness is asymmetry in a statistical distribution, in which the curve appears to be distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution. Here we have tried to show the skewness of our variables and we have found that our target variable absenteeism in hours having is one sided skewed so by using **log transform** technique we have tried to reduce the skewness of the same.

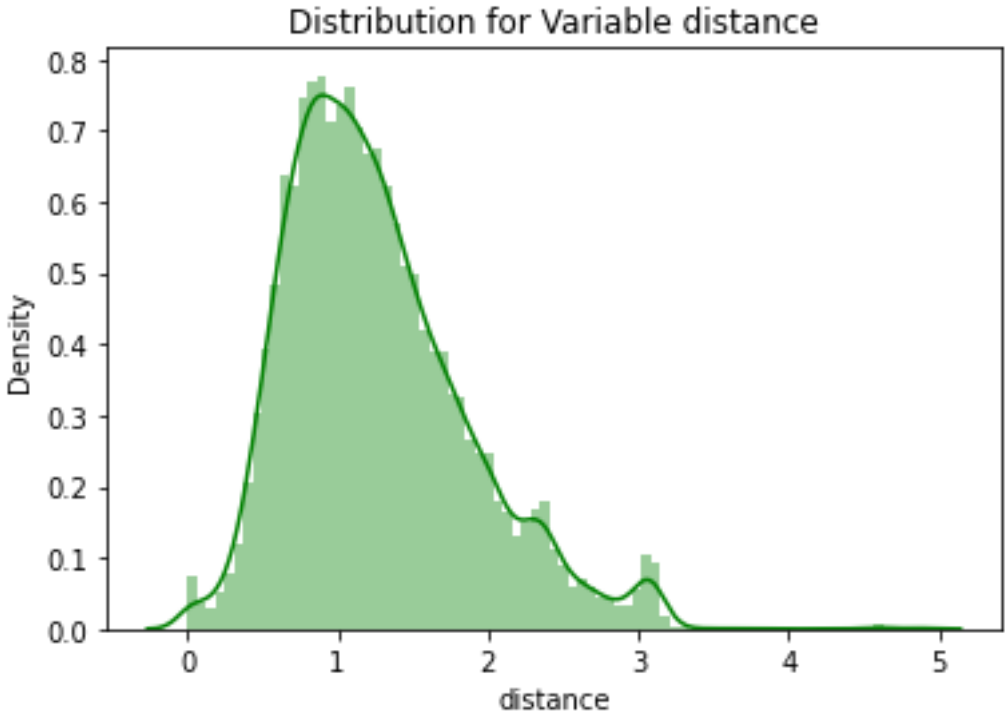
Below mentioned graphs show the probability distribution plot to check distribution before log transformation:





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





As our continuous variables appear to be normally distributed so we don't need to use feature scaling techniques like normalization and standardization for the same.

Chapter 4

Modelling

After a thorough preprocessing, we will use some regression models on our processed data to predict the target variable. Following are the models which we have built –

- Linear Regression
- Decision Tree
- Random Forest

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 our train data. Below is the snipped image of the split of train test.

We need to split our train data into two parts

```
In [56]: from sklearn.tree import DecisionTreeRegressor
         from sklearn.metrics import mean_squared_error

In [60]: ##train test split for further modelling
         X_train, X_test, y_train, y_test = train_test_split( train_df.iloc[:, train_df.columns != 'fare_amount'],
                                 train_df.iloc[:, 0], test_size = 0.20, random_state = 1)

In [61]: print(X_train.shape)
         print(X_test.shape)
```

4.1 Linear Regression

Multiple linear regression is the most common form of linear regression analysis. Multiple regression is an extension of simple linear regression. It is used as a predictive analysis, when we want to predict the value of a variable based on the value of two or more other variables. The variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable).

Below is a screenshot of the model we build and its output:

Linear Regression Model :

In [105]:
```
# Building model on top of training dataset
fit_LR = LinearRegression().fit(X_train , y_train)
```

In [106]:
```
#prediction on train data
pred_train_LR = fit_LR.predict(X_train)
```

In [107]:
```
#prediction on test data
pred_test_LR = fit_LR.predict(X_test)
```

Error & Accuracy

In [108]:
```
#https://stats.stackexchange.com/questions/11636/the-difference-between-mse-and-mape
#https://stats.stackexchange.com/questions/299712/what-are-the-shortcomings-of-the-mean-absolute-percentage-error-mape/299713#299713

def MAE(y_actual, y_predicted):
    mae=mean_absolute_error(y_actual, y_predicted)
    return "Mean Absolute Error is               :"+str(mae)

def RMSPE(y_actual, y_predicted):
    rmspe=(np.sqrt(np.mean(np.square((y_actual- y_predicted) /y_actual)))) * 100
    return "Root Mean Squared Percentage Error is :"+str(rmspe)

def RMSE(y_actual, y_predicted):
    mse=mean_squared_error(y_actual, y_predicted)
    rms = sqrt(mse)
    return "Root Mean Squared Error is           :"+str(rms)

def MAPE(y_actual, y_predicted):
    mape=mean_absolute_percentage_error(y_actual, y_predicted)*100
    return "Mean Absolute Percentage Error is    :"+str(mape)
```

In [109]:
```
#Train
```

In [110]:
```
display(MAE(y_train,pred_train_LR))
display(MAPE(y_train,pred_train_LR))
display(RMSE(y_train,pred_train_LR))
display(RMSPE(y_train,pred_train_LR))
```

```
'Mean Absolute Error is               :0.17281281199743248'

'Mean Absolute Percentage Error is    :7.55281760732123'

'Root Mean Squared Error is           :0.2662046515449016'

'Root Mean Squared Percentage Error is :11.783104342364096'
```

In [111]:
```
#Test
```

In [112]:
```
display(MAE(y_test,pred_test_LR ))
display(MAPE(y_test, pred_test_LR ))
display(RMSE(y_test, pred_test_LR ))
display(RMSPE(y_test,pred_test_LR ))
```

```
'Mean Absolute Error is               :0.17145022247394073'

'Mean Absolute Percentage Error is    :7.40667729514645'

'Root Mean Squared Error is           :0.2657932503002036'

'Root Mean Squared Percentage Error is :11.138925329198141'
```

4.2 Decision Tree

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

Below is the screenshot of the query we executed and the result shown, we will compare the results of each model in a combined table later on.

Decision Tree Algorithm

Decision tree Model :

In [113]:
```
fit_DT = DecisionTreeRegressor().fit(X_train,y_train)
```

In [114]:
```
#prediction on train data
pred_train_DT = fit_DT.predict(X_train)

#prediction on test data
pred_test_DT = fit_DT.predict(X_test)
```

In [115]:
```
#Train
```

In [116]:
```
display(MAE(y_train,pred_train_DT))
display(MAPE(y_train,pred_train_DT))
display(RMSE(y_train,pred_train_DT))
display(RMSPE(y_train,pred_train_DT))
```

```
'Mean Absolute Error is                :4.4819314710062256e-18'

'Mean Absolute Percentage Error is     :1.5385096272918867e-16'

'Root Mean Squared Error is            :5.069423008001435e-17'

'Root Mean Squared Percentage Error is :1.5503850053675335e-15'
```

In [117]:
```
#Test
```

In [118]:
```
display(MAE(y_test,pred_test_DT))
display(MAPE(y_test, pred_test_DT ))
display(RMSE(y_test, pred_test_DT ))
display(RMSPE(y_test,pred_test_DT ))
```

```
'Mean Absolute Error is                :0.2382984783714758'

'Mean Absolute Percentage Error is     :10.518250501599903'

'Root Mean Squared Error is            :0.3602069723698936'

'Root Mean Squared Percentage Error is :16.69816121405505'
```

4.3 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other task, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Below is a screenshot of the model we build and its output:

Random Forest Model :

In [119]:
```
fit_RF = RandomForestRegressor().fit(X_train,y_train)
```

In [120]:
```
#prediction on train data
pred_train_RF = fit_RF.predict(X_train)
#prediction on test data
pred_test_RF = fit_RF.predict(X_test)
```

In [121]:
```
##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))
```

In [122]:
```
#Train
```

In [123]:
```
display(MAE(y_train, pred_train_RF))
display(MAPE(y_train, pred_train_RF))#8     2
display(RMSE(y_train, pred_train_RF))#1     0
display(RMSPE(y_train, pred_train_RF))
```

```
'Mean Absolute Error is                :0.06276306089515532'

'Mean Absolute Percentage Error is     :2.8045313402636407'

'Root Mean Squared Error is            :0.09350403240633642'

'Root Mean Squared Percentage Error is :4.379061802387351'
```

In [124]:
```
#Test
```

In [125]:
```
display(MAE(y_test, pred_test_RF))
display(MAPE(y_test, pred_test_RF))#22    7
display(RMSE(y_test, pred_test_RF))#5     0
display(RMSPE(y_test, pred_test_RF))
```

```
'Mean Absolute Error is                :0.16539068086455702'

'Mean Absolute Percentage Error is     :7.3274806151377065'

'Root Mean Squared Error is            :0.24715958202085736'

'Root Mean Squared Percentage Error is :11.104469906499823'
```

Optimizing the results with parameters tuning :

In [126]:
```
# Look at parameters used by our current forest
print('Parameters currently in use:\n')
display(fit_RF.get_params())
```

```
Parameters currently in use:


{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

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.

Random Hyperparameter Grid

In [127]:
```
##Random Search CV on Random Forest Model

RRF = RandomForestRegressor()
n_estimator = list(range(100,200,10))
depth = list(range(1,100,2))

# Create the random grid
rand_grid = {'n_estimators': n_estimator,'max_depth': depth}


randomcv_rf = RandomizedSearchCV(RRF, param_distributions = rand_grid, cv = 5)
randomcv_rf = randomcv_rf.fit(X_train,y_train)
predictions_RRF1 = randomcv_rf.predict(X_train)
predictions_RRF2 = randomcv_rf.predict(X_test)
```

In [128]:
```
view_best_params_RRF = randomcv_rf.best_params_
display(view_best_params_RRF)
best_model = randomcv_rf.best_estimator_
display(best_model)
```

```
{'n_estimators': 110, 'max_depth': 13}

RandomForestRegressor(max_depth=13, n_estimators=110)
```

In [129]:
```
#Train
```

In [130]:
```
display(MAE(y_train, predictions_RRF1))
display(MAPE(y_train,predictions_RRF1))
display(RMSE(y_train,predictions_RRF1))
display(RMSPE(y_train,predictions_RRF1))
```

```
'Mean Absolute Error is                :0.09970708240188524'

'Mean Absolute Percentage Error is     :4.527920275103174'

'Root Mean Squared Error is            :0.1369053679851192'

'Root Mean Squared Percentage Error is :6.34839766411581'
```

In [131]:
```
#Test
```

In [132]:
```
display(MAE(y_test,predictions_RRF2))
display(MAPE(y_test,predictions_RRF2))
display(RMSE(y_test,predictions_RRF2))
display(RMSPE(y_test,predictions_RRF2))
```

```
'Mean Absolute Error is                :0.16329296576861832'

'Mean Absolute Percentage Error is     :7.216493269066902'

'Root Mean Squared Error is            :0.24672449257206835'

'Root Mean Squared Percentage Error is :11.054734501570492'
```

Here we have used hyper parameter tuning techniques follows:

- Random Search CV

1. **Random Search CV**: This algorithm set up a grid of hyperparameter values and select random combinations to train the model and score. The number of search iterations is set based on time/resources.

```
In [134]: predictions_test=randomcv_rf.predict(test)

In [135]: test['Predicted_fare'] = predictions_test
```

Chapter 5

Conclusion

5.1 Model Evaluation

The main concept of looking at what is called residuals or difference between our predictions f(x[I,]) and actual outcomes y[i].

In general, most data scientists use two methods to evaluate the performance of the model:

I. **RMSE** (Root Mean Square Error): is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modelled.

$$RMSE = \sqrt{\frac{\sum_{1}^{n} (X_{obs,i} - X_{model,i})^2}{n}}$$

https://stackoverflow.com/questions/53165807/how-to-**calculate-rmspe**-in-python-using-numpy

II. MAE (Mean absolute error) represents the difference between the original and predicted values extracted by averaged the absolute difference over the data set.

III. Mean Absolute Percentage Error (MAPE) is a statistical measure to define the accuracy of a machine learning algorithm on a particular dataset. MAPE can be considered as a loss function to define the error termed by the model evaluation.

IV. We have shown both train and test data results, the main reason behind showing both the results is to check whether our data is overfitted or not.

Below table shows the model results before applying hyper tuning:

Model Name	**MAE**		**MAPE**		**RMSE**		**RMSPE**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**

Linear Regression	0.172	0.171	7.55	7.40	0.26	0.26	11.7	11.1
Decision Tree	4.48	0.23	1.53	10.51	5.06	0.36	1.55	16.69
Random Forest model	0.06	0.16	2.80	7.32	0.09	0.24	4.37	11.1

Below table shows results post using hyper parameter tuning techniques:

Model Name		**MAE**	**MAPE**	**RMSE**	**RMSPE**
Random Search CV	Random Forest	0.1 0.1	4.5 7.2	0.1 0.2	6.3 11.05

Above table shows the results after tuning the parameters of our best suited model i.e. Random Forest . For tuning the parameters, we have used Random Search CV under which we have given the range of n_estimators and depth .

5.2 Model Selection

RMSE is scale-dependent, MAPE is not.
So if we are comparing accuracy across time series with different scales, we can't use RMSE.
For business use, MAPE is often preferred because apparently managers understand percentages better than squared errors.
MAPE can't be used when percentages make no sense.
For example, the Fahrenheit and Celsius temperature scales have relatively arbitrary zero points, And it makes no sense to talk about percentages.
MAPE also cannot be used when the time series can take zero values.

Finally we can say -
MAPE is not scale dependent.
The value of fare and distance can never be zero so we are choosing MAPE instead of RMSE.

On the basis RMSE should have least value and MAPE should have lest percentage error. So, from above tables we can see:

- From the observation of all RMSE Value and MAPE Value we have concluded that,
- Both the models- Decision Tree and Random Forest perform comparatively well While comparing their RMSE and MAPE value.
- After this, I chose Random Forest CV to apply cross validation technique and see changes brought about by that.
- After applying tunings Random forest model shows the best result..

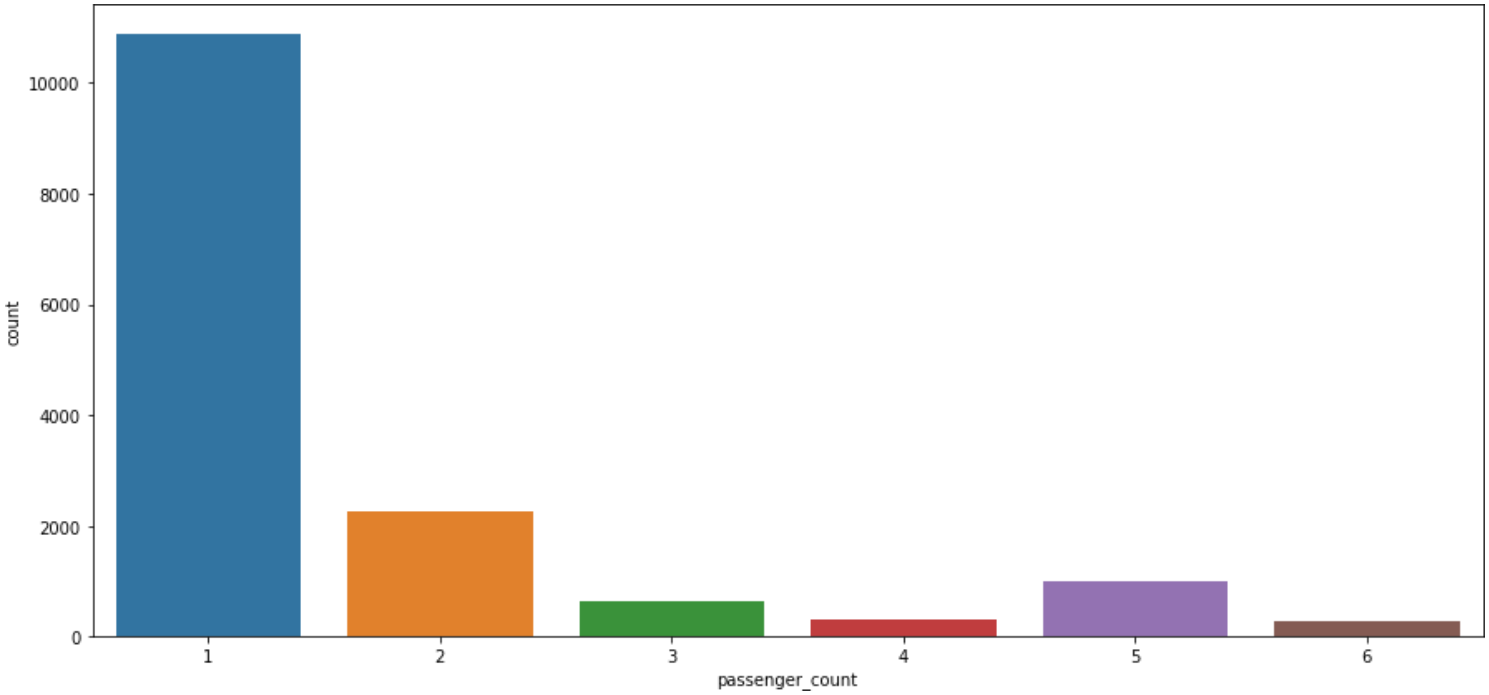
- So finally, we can say that Random forest model is the best method to make prediction for this project with highest explained variance of the target variables and lowest error chances with parameter tuning technique Random Search CV.

Finally, I used this method to predict the target variable for the test data file shared in the problem statement. Results that I found are attached with my submissions.

5.3 Some more visualization facts:

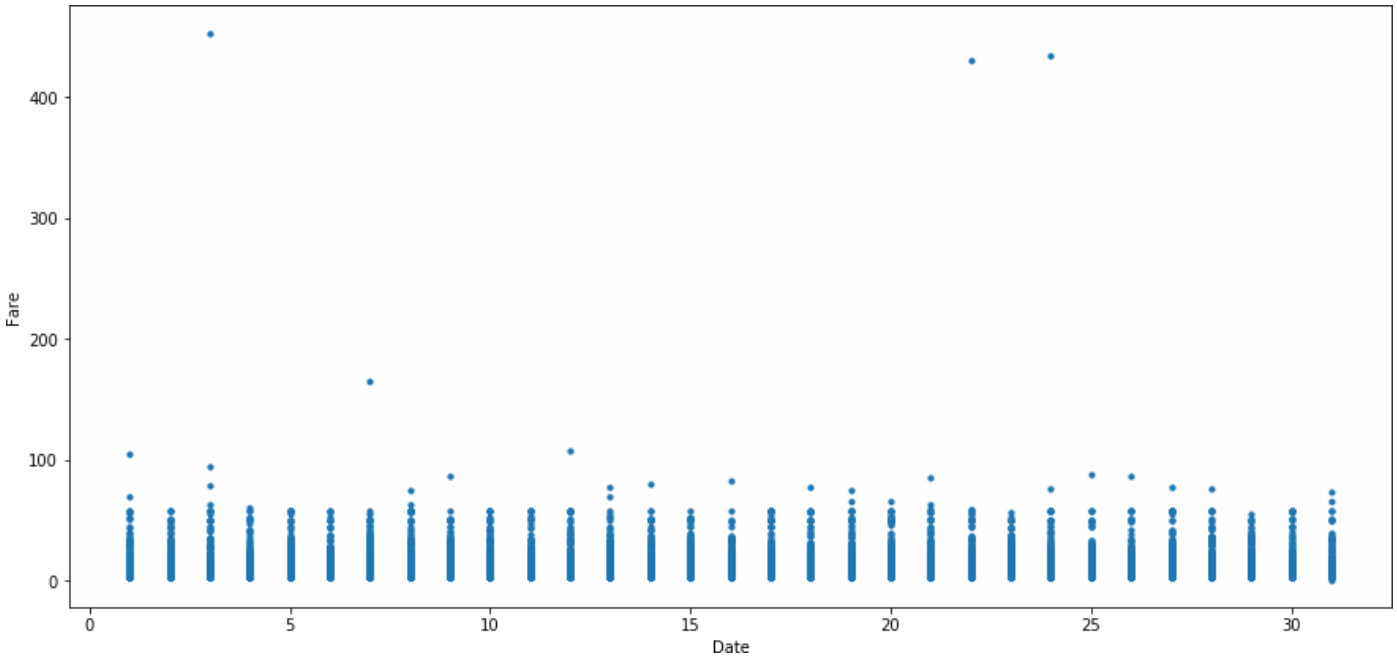
1. **Number of passengers and fare**

   We can see in below graph that single passengers are the most frequent travelers, and the highest fare also seems to come from cabs which carry just 1 passenger.

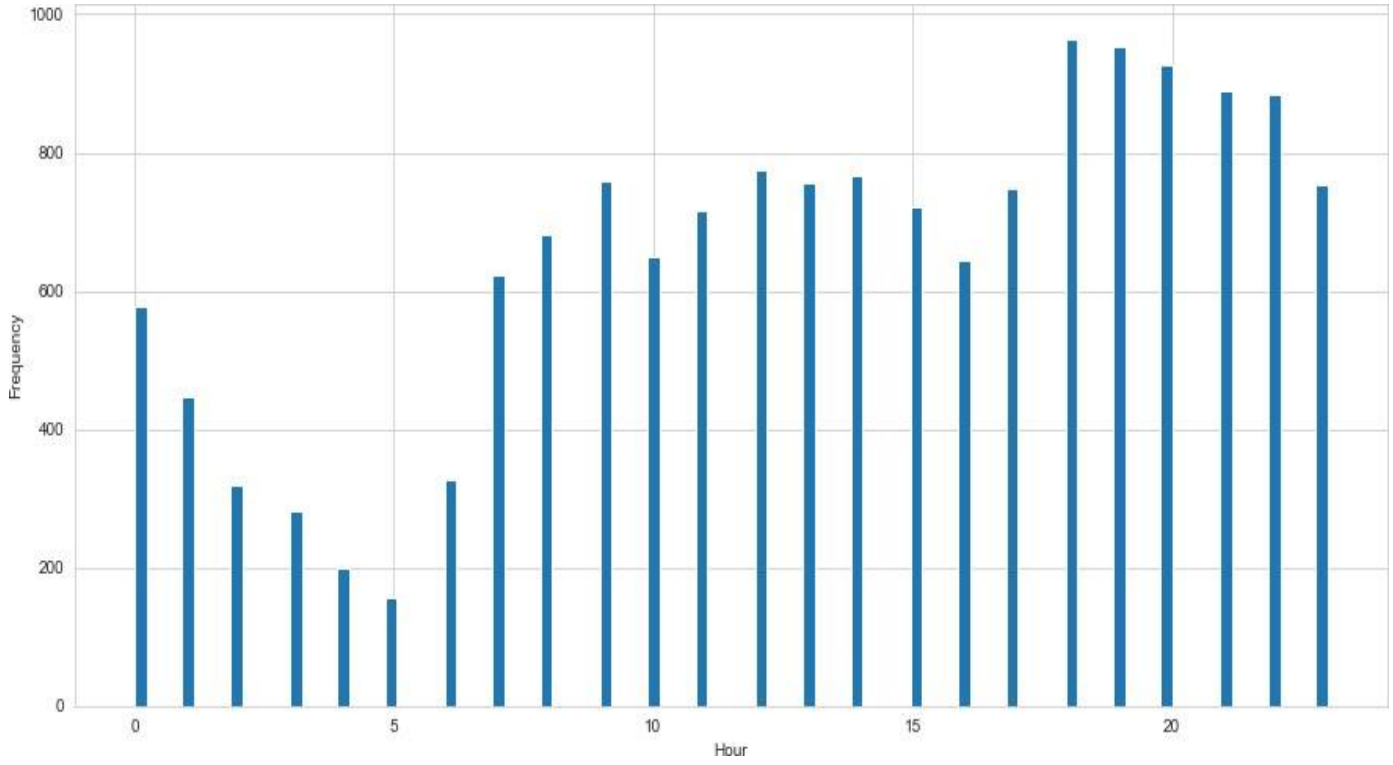

10000
8000
6000
4000
2000
0
count
1
2
3
4
5
6
passenger_count

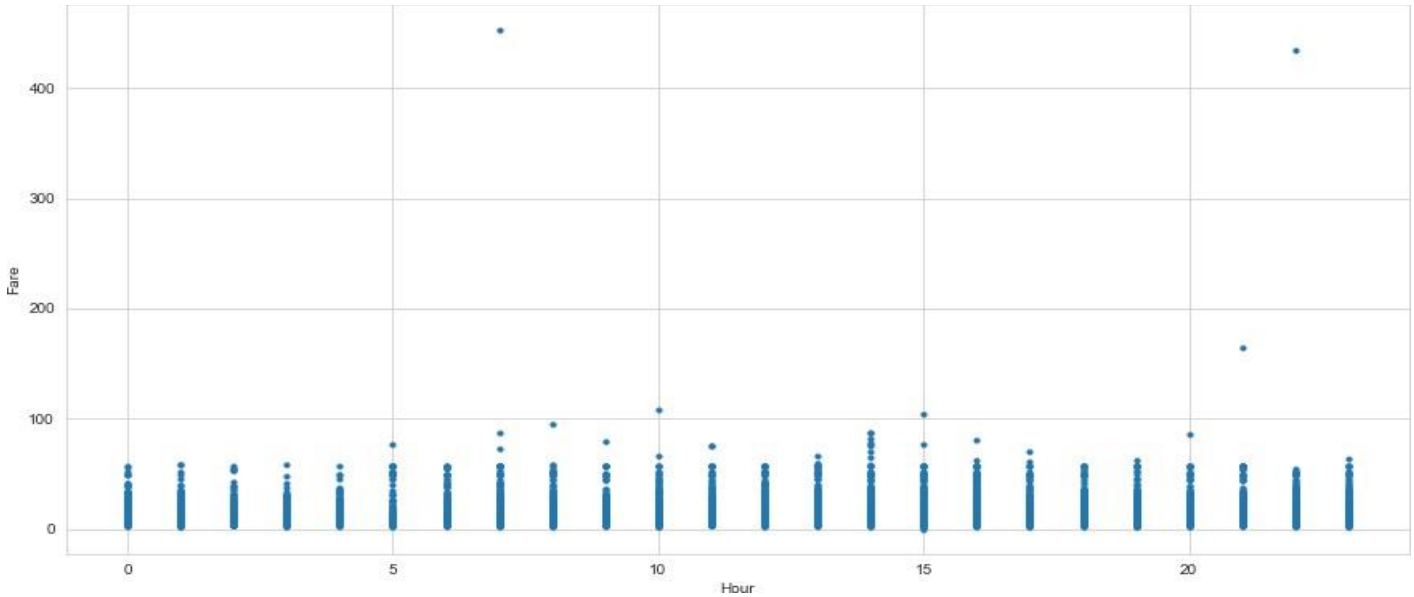
2. Date of month and fares

The fares throughout the month mostly seem uniform.



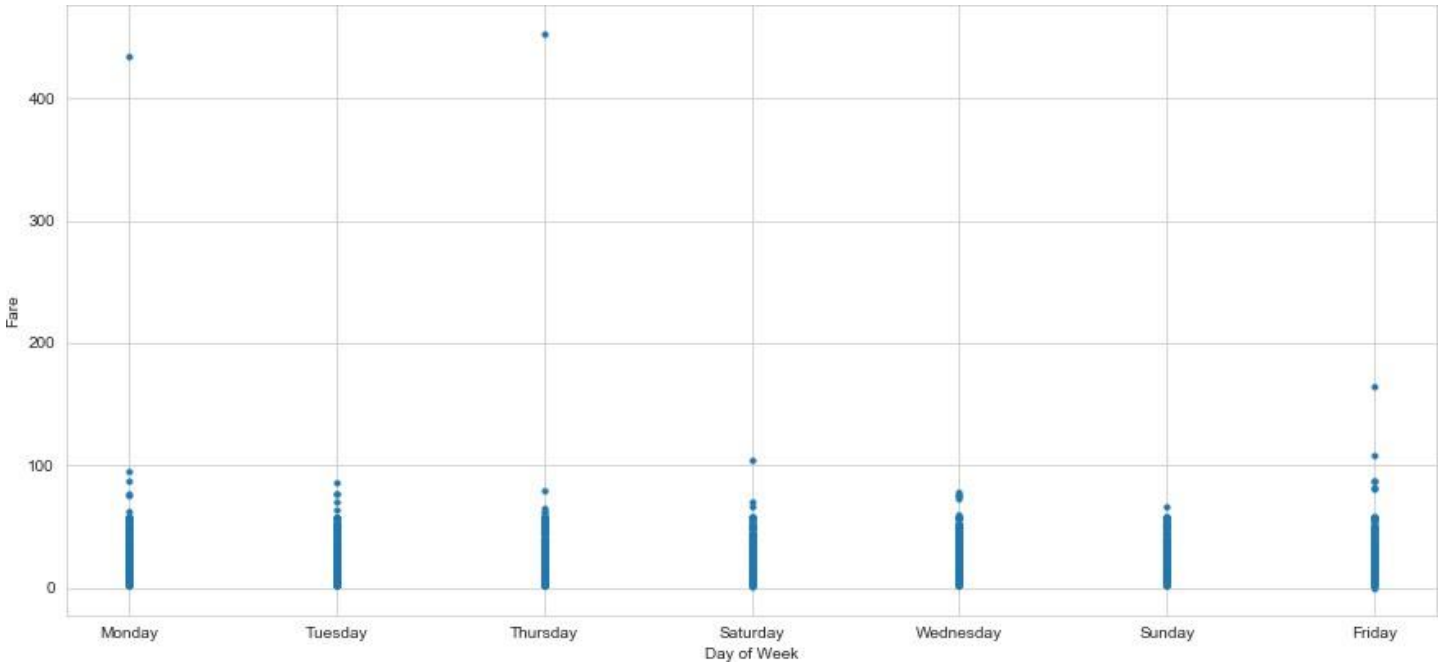
3. **Hours and Fares**
    - During hours 6 PM to 11PM the frequency of cab boarding is very due to peak hours
    - Fare prices during 2PM to 8PM is bit high compared to all other time might be due to high demands.



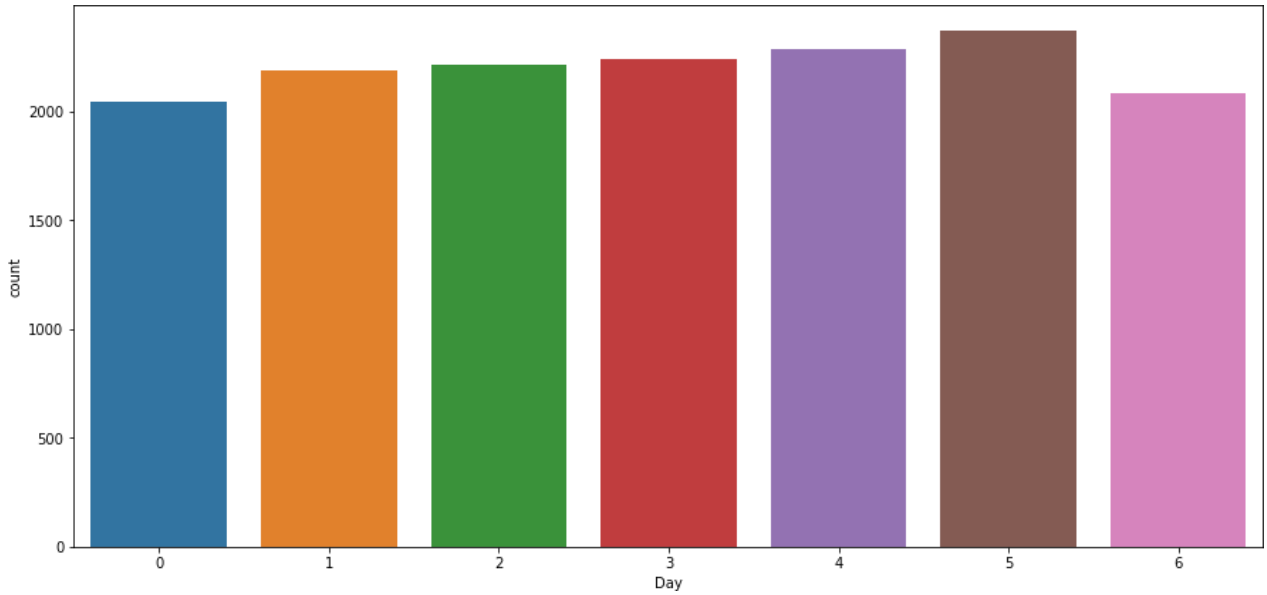


4. **Week Day and fare**

- Cab fare is high on Friday, Saturday and Monday, may be during weekend and first day of the working day they charge high fares because of high demands of cabs.



5. Impact of Day on the Number of Cab rides :



Observation : The day of the week does not seem to have much influence on the number of cabs ride