Cab Fare Prediction Report

Author: Abhishek Goswami

11/09/2019

Contents

Chapter 1

Introduction:

Nowadays cab rental services are expanding with the multiplier rate. 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:

Understanding of data is the very first and important step in the process of finding solution of any business problem. Here in our case we were provided with dataset with following features, we need to go through each and every variable of it to understand and 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

  Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots.
  This is often called as Exploratory Data Analysis. EDA involves some of the steps below mentioned.
    - Data exploration and Cleaning
    - Missing value treatment
    - Outlier Analysis
    - Feature Selection
    - Features Scaling
    - Visualization
    - Skewness and Log transformation

- ### Modelling

  Once all the Pre-Processing steps has been done on our data set, we will now further move to our next step which is modelling. Modelling plays an important role to find out the good inferences from the data. Choice of models depends upon the problem statement and data set. 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 following models need to be tested:

    - Linear regression
    - Decision Tree
    - Random forest

- ✓ We have also used hyper parameter tunings to check the parameters on which our model runs best. Following is the techniques of hyper parameter tuning we have used:
    - Grid Search CV

❖ Model Selection

The final step of our methodology will be the selection of 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 following points as per this project:

1. Separate the combined variables.
2. As we know we have some negative values in fare amount so we have to remove those values.
3. Passenger count would be max 6 if it is a SUV vehicle not more than that. We have to remove the rows having passenger_count more than 6 and less than 1.
4. There are some outlier figures in the fare (like fares > 150) so we need to remove them.
5. 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 mentioned range.

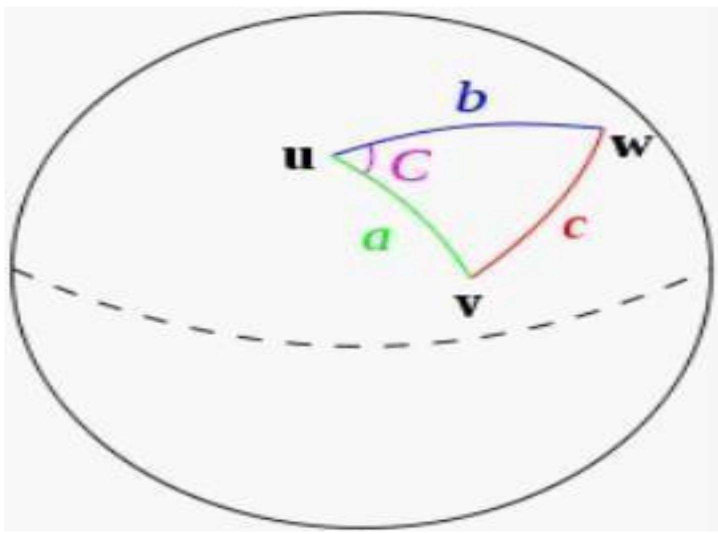
3.2 Creating some new variables out of the given variable.

Here in our data set our variable name pickup_datetime contains date and time for pickup which is of type timestamp. So we tried to extract some important variables from pickup_datetime:

- Pickup_month
- Pickup_date
- Pickup_day (0 is Monday and 6 is Sunday in python and 1 is Sunday and 7 is Saturday in R)
- Pickup_Hour (24 denotes 00: i.e. 12:00 am in python and 0 denotes 12:00 am in R)
- Pickup_Minute

Also, we 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
- pickup_month
- pickup_date
- pickup_day
- pickup_hour
- pickup_minute
- distance_travelled

3.3 Selection of variables

Now as we have extracted the meaningful information from the given variables so we will drop the redundant variables which are as follows:

- pickup_datetime
- pickup_longitude
- pickup_latitude
- dropoff_longitude
- dropoff_latitude

Now only following variables we will use for further steps:

Index	fare_amount	passenger_count	pickup_date	pickup_day	pickup_month	pickup_hour	pickup_minute	distance_travelled
0	4.500000	1.000000	15.000000	0.000000	6.000000	17.000000	26.000000	1.030764
1	16.900000	1.000000	5.000000	1.000000	1.000000	16.000000	52.000000	8.450134
2	5.700000	2.000000	18.000000	3.000000	8.000000	24.000000	35.000000	1.389525
3	7.700000	1.000000	21.000000	5.000000	4.000000	4.000000	30.000000	2.799270
4	5.300000	1.000000	9.000000	1.000000	3.000000	7.000000	51.000000	1.999157
5	12.100000	1.000000	6.000000	3.000000	1.000000	9.000000	50.000000	3.787239
6	7.500000	1.000000	20.000000	1.000000	11.000000	20.000000	35.000000	1.555807
7	16.500000	1.000000	4.000000	2.000000	1.000000	17.000000	22.000000	4.155444
9	8.900000	2.000000	2.000000	2.000000	9.000000	1.000000	11.000000	2.849627

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, pickup_minute, pickup_month, pickup_date, pickup_day, pickup_hour, distance_travelled.

Our Dependent variable is: **fare_amount**

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

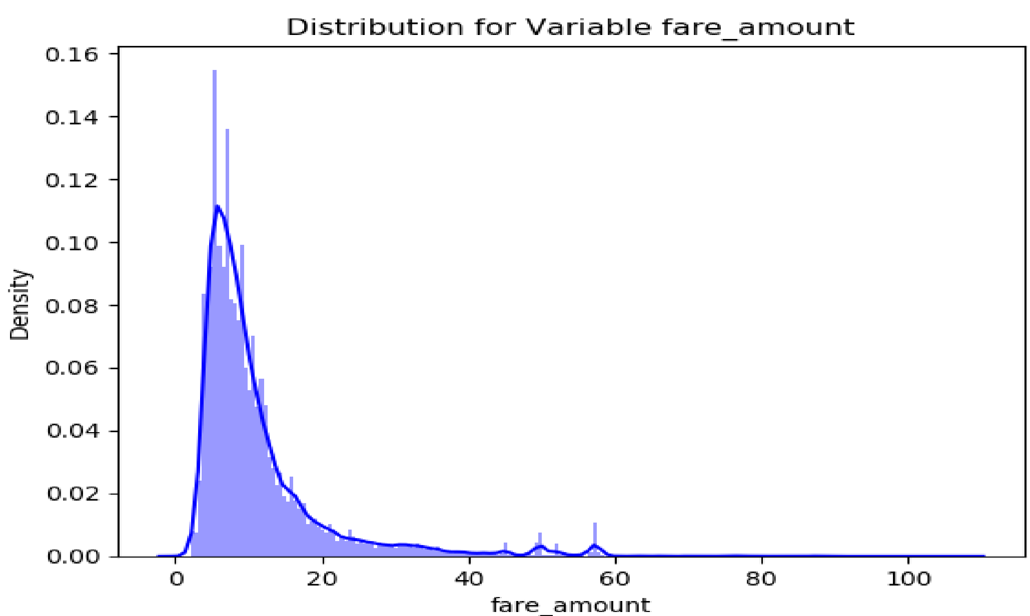
Continuous variables - 'fare_amount', 'distance_travelled'

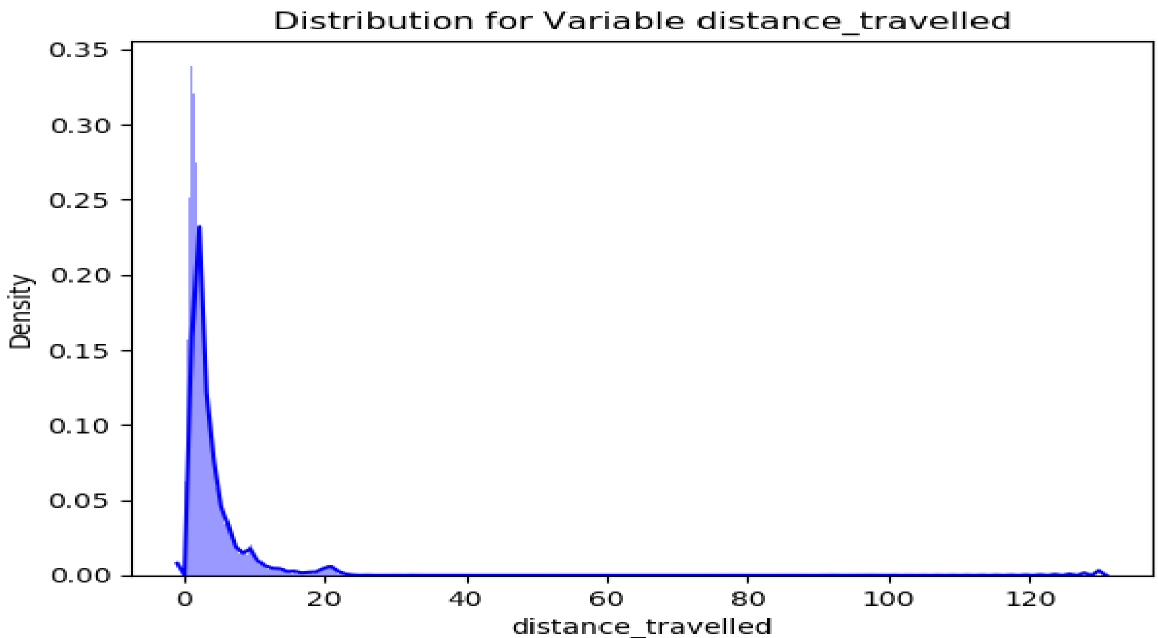
Categorical Variables - 'pickup_Month', 'pickup_Date', 'pickup_weekday', 'pickup_hour', 'passenger_count'

3.5 Feature Scaling

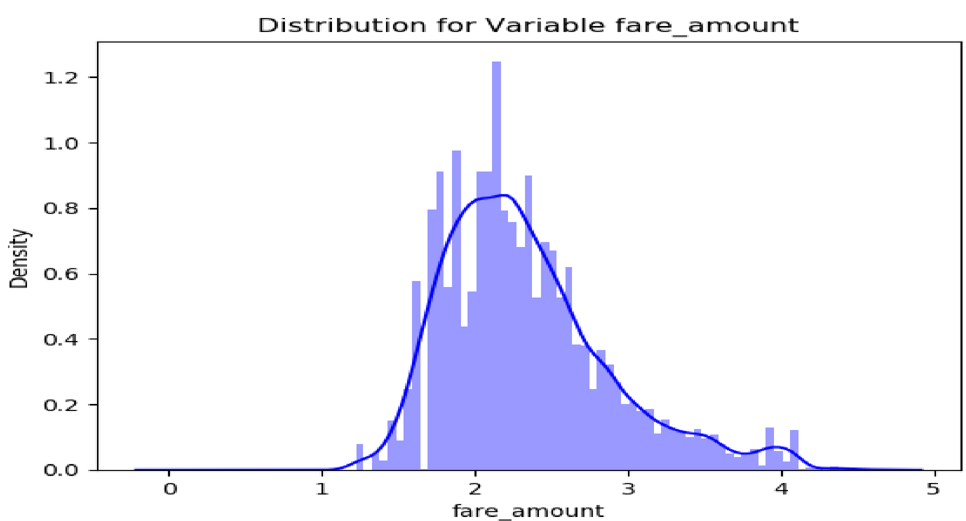
Skewness is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution. Here we tried to show the skewness of our variables and we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

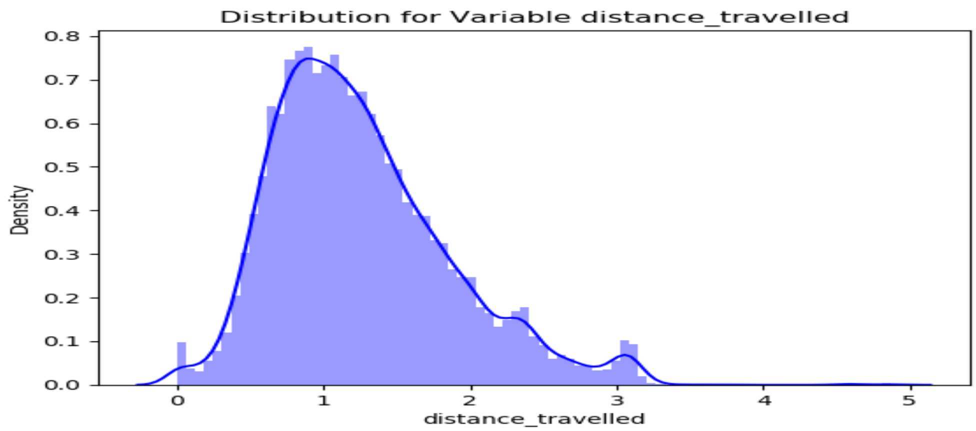
Below mentioned graphs shows the probability distribution plot to check distribution before log transformation:





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





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

Chapter 4

Modelling:

In this case we have to predict the cab fares based on given set of predictors. So, the target variable here is a continuous variable. For Continuous we can use various Regression models. Model having less error rate and more accuracy will be our final model.

Models built are: -

- c50 (Decision tree for regression target variable)
- Random Forest (with 300 trees)
- Linear regression

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 dataset as our train data. Below is the snipped image of the split of our dataset into train set and test set.

```
# Seperating independent and dependent variables.
X = dataset.iloc[:, 1:].values
Y = dataset.iloc[:, 0].values

# SPLITTING THE DATA INTO TRAIN AND TEST.
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)

```

4.1 Linear Regression

It is one of the most widely known modeling technique. Linear regression is usually among the first few topics which people pick while learning predictive modeling. In this technique, the dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear.

Creating Model: -

In R:

```
set.seed(123)
split = sample.split(dataset$fare_amount, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Building Models

# 1. Multiple Linear Regression Model
regressor_LR = lm(formula = fare_amount ~ .,
               data = training_set)
summary(regressor_LR)

# Predicting the test set
y_pred = predict(regressor_LR, newdata = test_set)
```

Environment

Files Plots Packages Help Viewer

Zoom Export

In Python:

```
# Building Models

# 1. Multiple Linear Regression Model
from sklearn.linear_model import LinearRegression
regressor_LR = LinearRegression()
regressor_LR.fit(X_train, Y_train)
# Predicting on Test Set
Y_pred = regressor_LR.predict(X_test)
```

4.2 Decision Tree

This model is also known as Decision tree for regression target variable.
For this model we have divided the dataset into train and test part using random sampling. Where train contains 80% data of dataset and test contains 20% data of dataset. Decision Tree is a non linear and non-continuous model.

Creating Model: -

In R:

```r

# 2. Decision Tree Model
library(rpart)
regressor_DT = rpart(formula = fare_amount ~ .,
                     data = training_set,
                     control = rpart.control(minsplit = 7))
summary(regressor_DT)

# Predicting the test set
y_pred = predict(regressor_DT, newdata = test_set)

```

In python:

```python

# 2. Decision Tree
# Fitting Decision Tree Regression Model
from sklearn.tree import DecisionTreeRegressor
regressor_DT = DecisionTreeRegressor(max_depth = 10, random_state = 0)
regressor_DT.fit(X_train, Y_train)

# Predicting on Test Set
Y_pred = regressor_DT.predict(X_test)

```

4.3 Random Forest

Random forests or random decision forests is an ensemble learning method for classification, regression and other task, which operates 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 all the individual trees.

To say it in simple words: Random forest builds multiple decision trees and Outputs the prediction voted by all the individual tress

Creating Model: -

In R:

```r
# 3. Random Forest Model
# Fitting Random Forest Regression Model to the model
library(randomForest)
set.seed(1234)
regressor_RF = randomForest(x = training_set[, 2:7],
                            y = training_set$fare_amount,
                            ntree = 500)
# Predicting the test set
y_pred = predict(regressor_RF, newdata = test_set)
```

In Python:

```python
# 3. Random Forest
from sklearn.ensemble import RandomForestRegressor
regressor_RF = RandomForestRegressor(max_depth = 7, n_estimators = 300, random_state = 1)
regressor_RF.fit(X_train, Y_train)

# Predicting on Test Set
Y_pred = regressor_RF.predict(X_test)

```

Chapter 5

Conclusion

5.1 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 Predictions, 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.

Mean Absolute Percentage Error (MAPE)

MAPE is one of the error measures used to calculate the predictive performance of the model. We will apply this measure to our models that we have generated in our project.

```
# Calculating Mape
def MAPE(true, pred):
    mape = np.mean(np.abs((true - pred)/true))* 100
    return mape
```

In above function 'true' is the actual value and 'pred' is the predicted value. It will provide the error percentage of model.

Root Mean Squared Error (RMSE)

RMSE is the most popular evaluation metric used in regression problems. It follows an assumption that the errors are unbiased and follow a normal distribution.

```python

def RMSE(true, pred):
    mse = np.mean((true - pred)**2)
    print('Mean Square : ', mse)
    rmse = np.sqrt(mse)
    print('Root Mean Square :', rmse)
```

MAPE, RMSE and R-squared values for different models in Python are as follows:

6. Linear Model

```
   ...:
   ...: MAPE(Y_test, Y_pred)
Out[3]: 12.836523925164014

In [4]: from sklearn import metrics
   ...: print(np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))
0.2900719644831155

In [5]: from sklearn.metrics import r2_score

In [6]: r2_score(Y_test, Y_pred)
Out[6]: 0.7350213834796294
```

7. Decision Tree Model

```
In [8]: Y_pred = regressor_DT.predict(X_test)

In [9]: MAPE(Y_test, Y_pred)
Out[9]: 14.791310968744764

In [10]: print(np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))
0.299728047403293

In [11]: r2_score(Y_test, Y_pred)
Out[11]: 0.717086230746008
```

8. Random Forest Model

```
In [15]: MAPE(Y_test, Y_pred)
Out[15]: 13.713597905431785

In [16]: print(np.sqrt(metrics.mean_squared_error(Y_test, Y_pred)))
0.2663340088977002

In [17]: r2_score(Y_test, Y_pred)
Out[17]: 0.7766157402514353
```

MAPE and RMSE values in R:

1. Linear Model

```
     rmse      mape
0.2723072 0.0772204
>
```

2. Decision Tree Model

```
      rmse       mape
0.27760637 0.08594868
>
```

3. Random Forest

```
      rmse       mape
0.26185654 0.08085457
>
```

5.2 Model Selection

We can see that in both R and Python Random Forest Model fits the best out of Decision Tree and Linear Regression. RMSE Value is the lowest for Random Forest. So, to improve the model and enhance its performance so that it can perform efficiently on new test set, implemented **K Fold Cross Validation**.

In Python-

```
# Applying K-Fold Cross Validation for Random Forest
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor_RF,
                             X = X_train,
                             y = Y_train,
                             cv = 7)
```

After applying K Fold Cross Validation on Random Forest We did hyper tunning using grid search cv.

Grid Search CV: This algorithm sets up a grid of hyperparameter values and for each combination, trains a model and checks score on the validation data. In this approach, every single combination of hyperparameters values is tried which can be very inefficient.

Grid Search CV on Random forest

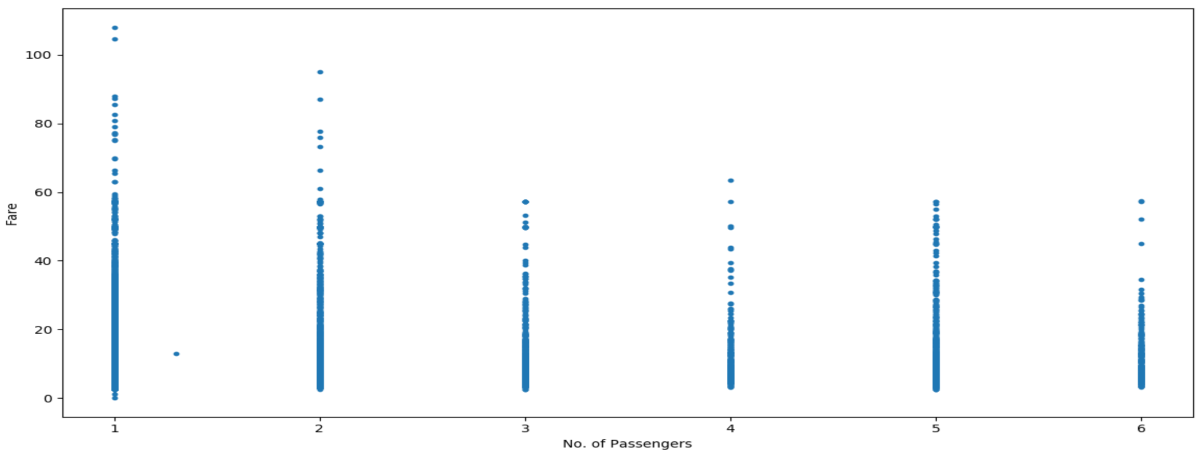
```

# Grid Search
from sklearn.model_selection import GridSearchCV
parameters = [{'max_depth' : [5,7,9], 'n_estimators' : [300, 400, 500],
               'random_state' : [0,1,2]}]
grid_search = GridSearchCV(estimator = regressor_RF,
                           param_grid = parameters,
                           cv = 7,
                           n_jobs = -1)
grid_search = grid_search.fit(X, Y)

# getting the scores and parameters
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
```

When we check the best_parameters, we find that the best values for the hyperparameters ‘max_depth’ and ‘n_estimators’ are 7 and 300 respectively.

These values of hyperparameters have been applied to the random forest model.

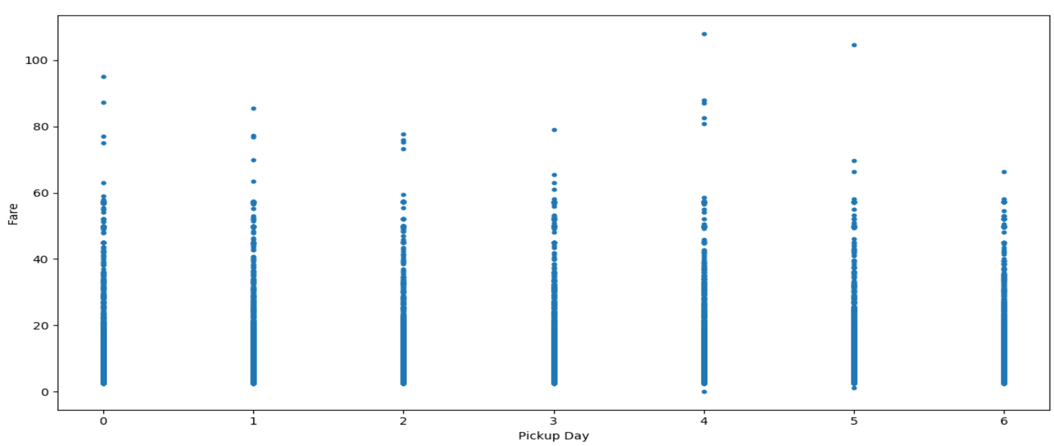
5.3 Visualization

1. passenger_count Vs fare_amount



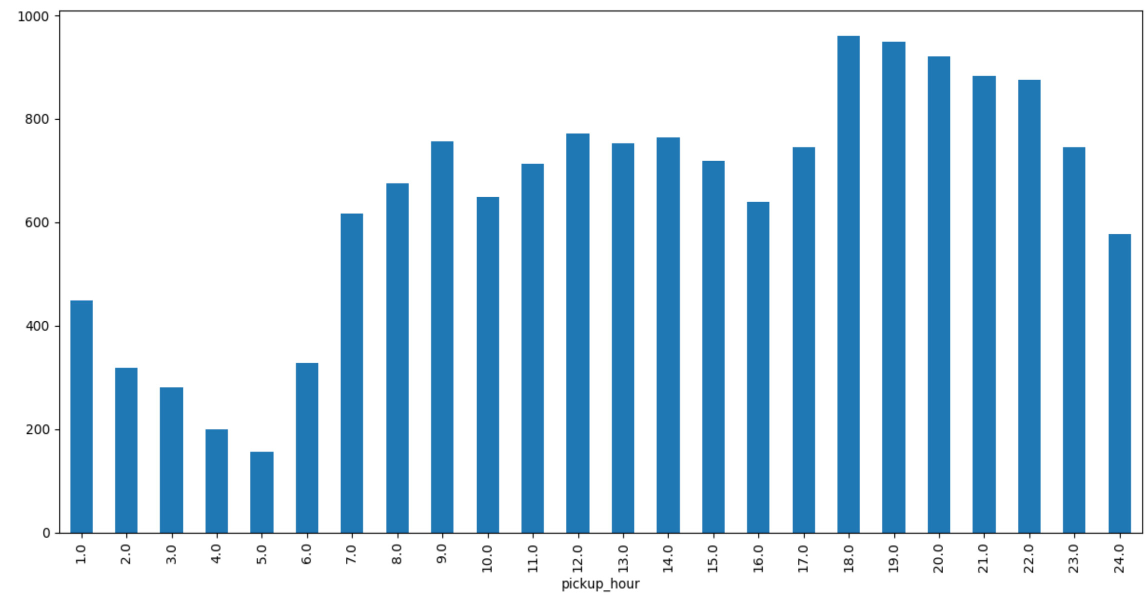
We could conclude that passengers who are travelling alone and rides with two travelers are contributing more.

2. Pickup_day Vs fare_amount



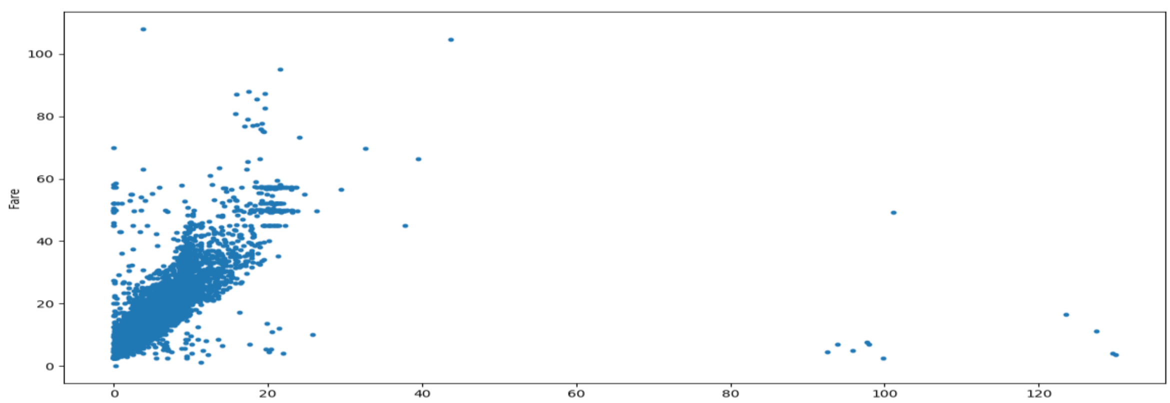
We could see that fares are higher on Monday (0), Tuesday (1), Wednesday (2), Friday (4) and Saturday (5) probably because of higher demands.

3. Pickup_hour



We could conclude that cab demands are more between 18:00 pm to 23:00 pm and least demand between 2:00 am to 5:00 am.

4. Distance_travelled Vs fare_amount



We could see that as distance increases fare amount increases. However, some very low fares for distance near 100km and distance more than 120km either they could be some very high discounted rides or noise in our dataset.

END OF REPORT