**PROJECT REPORT ON CAB FARE PREDICTION**

**BY USHA RAJU**

**INDEX**

1. Introduction

1.1 Problem Statement

1.2 Data

2. Methodology

2.1 Data Exploration

* Identification of variables and their datatypes
* Descriptive statistics
* Conversion of data types into required ones
* Missing Value Analysis
* Outlier Analysis
* Feature Engineering
* Data visualization

2.2 Modelling

* + - Model Development
      * + Linear Regression
        + Decision Tree
        + Random Forest
        + Gradient Boosting
* Model Evaluation
* Model Optimization
* Hyper Parameters Tunings for optimizing the results
* Model Selection

2.3 Conclusion

**1. Introduction**

Now a day’s 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 our company has provided a data set 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 Present: Yes

Outliers Present: 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 and 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 travelling in the cab |

**2 .Methodology**

**2.1 Data Exploration**

When required to build a predictive model, we need to understand the raw data before we start developing a model. This includes multiple pre-processing steps such as:

1. Identification of variables and their datatypes
2. Descriptive statistics
3. Conversion of data types into required ones
4. Missing Value Analysis
5. Outlier Analysis
6. Feature Engineering

**1. Identification of variables and their datatypes:**

* To get the dimensions of training and test dataset.
* To identify the target and predictor variables present in the dataset.
* To get the datatypes of the variables present.

**2. Descriptive statistics:**

* To get the not null entries in training and test datasets.
* To get the summary of variables from training and test datasets. This includes count, mean, standard deviation, minimum and maximum values and the quantiles of the data.
* To get the unique values and count of unique values from the datasets.

**3. Conversion of data types of variables into required ones:**

* To Convert the datatype of a variable into its required type in training and test datasets.
* passenger\_count – converted to categorical variable because passenger\_count cannot take continuous values as they are limited to the number of passengers in a cab.
* fare\_amount – converted to numeric using pd.to\_numeric() function with errors =” coerce” from object datatype.
* pickup\_datetime – converted to datetime datatype from object using pd.to\_datetime() function.

**4. Missing value Analysis:**

Missing or null values have to be eliminated from the datasets as it will impact in the performance of the model developed.

* To get the sum of null values from training and test datasets.
* To create dataframe to analyze missing values present.
* Since the percentage of missing values of passenger\_count and fare\_amount in training dataset is less, we can impute the missing values statistically.
* There are no missing values present in test dataset.
* Imputation of missing values in training dataset:

Imputation is a method to fill in the missing values with estimated ones. Mean / Mode / Median imputation is one of the most frequently used methods. It consists of replacing the missing data for a given attribute by the mean or median (quantitative attribute) or mode (qualitative attribute) of all known values of that variable.

* Since, fare\_amount and passenger\_count are numerical and categorical, we can opt for Mean/Median/Mode and KNN methods for imputation. To find the apt method out of all,
  + Remove any random value from the column missing and replace with NA.
  + Impute the null value with methods one by one.
  + Select the method with output nearly equal to the original value.

**5. Outlier Analysis:**

Outliers are the values that are not in the desired range. Based on the basic understanding of the data from Descriptive statistics, it is obvious that the outliers present in the training data are due to incorrectly entered or measured data. Hence, we can remove those values.

* Pickup and Dropoff latitudes and longitudes:
  + pickup and dropoff latitude should be in the range of -90 to +90 and pickup and dropoff longitude should be in the range of -180 to +180. Hence, rows having pickup and dropoff latitude, longitude values out of the given range should be considered as outliers.
* Passenger count:
  + Maximum number of passengers that can travel in a car can be only 6. Hence, values greater than 6 are outliers.
  + Passenger count cannot be 0. Hence, any value less than or equal to zero is an outlier.
* Fare\_amount:
  + Fare amount can't be zero or less than that. Hence, values less than or equal to 0 are outliers.

Now the values of the variables are in between maximum and minimum ranges.

**6. Feature Engineering:**

**Feature Creation:**

**1. Distance:**

We have latitude and longitude coordinates. Thus, we can calculate distance between the coordinates. This can be done based on haversine formula. The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Here, we create a new derived feature - distance.

After calculating distance and while observing the values, we can conclude that:

1. Distance values greater than 130 could be considered as outliers and removed.
2. Distance cannot be zero. The rows containing distance equal to 0 can also be dropped.

**2. Year, Month, Day, Date, Hour:**

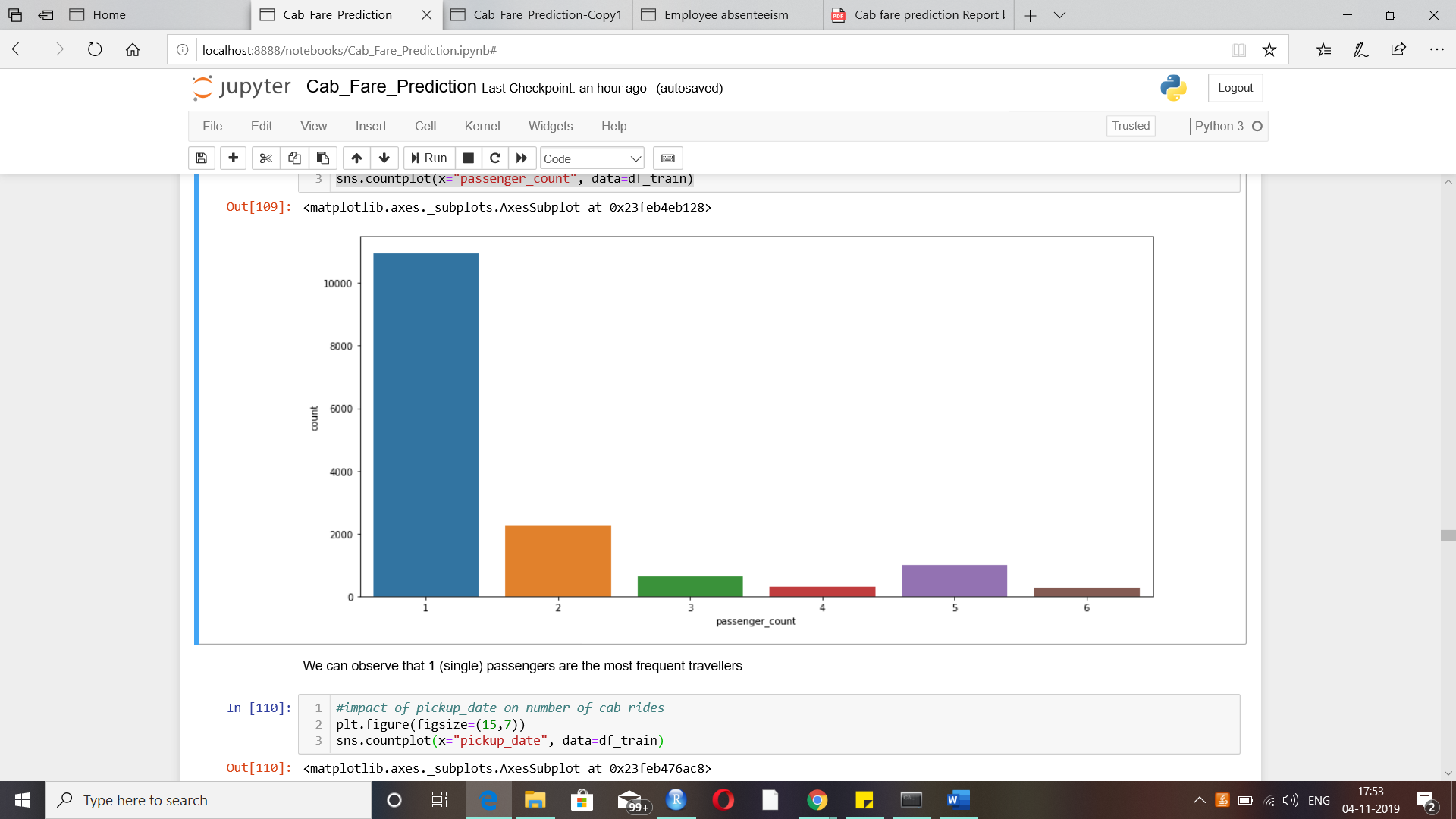
Now, we have pickup\_datetime feature with us. We can generate new variables like year,

month, date, day, hour from it which might have better relationship with our target variable. This also helps to highlight the hidden relationship in a variable.

**Data Visualization:**

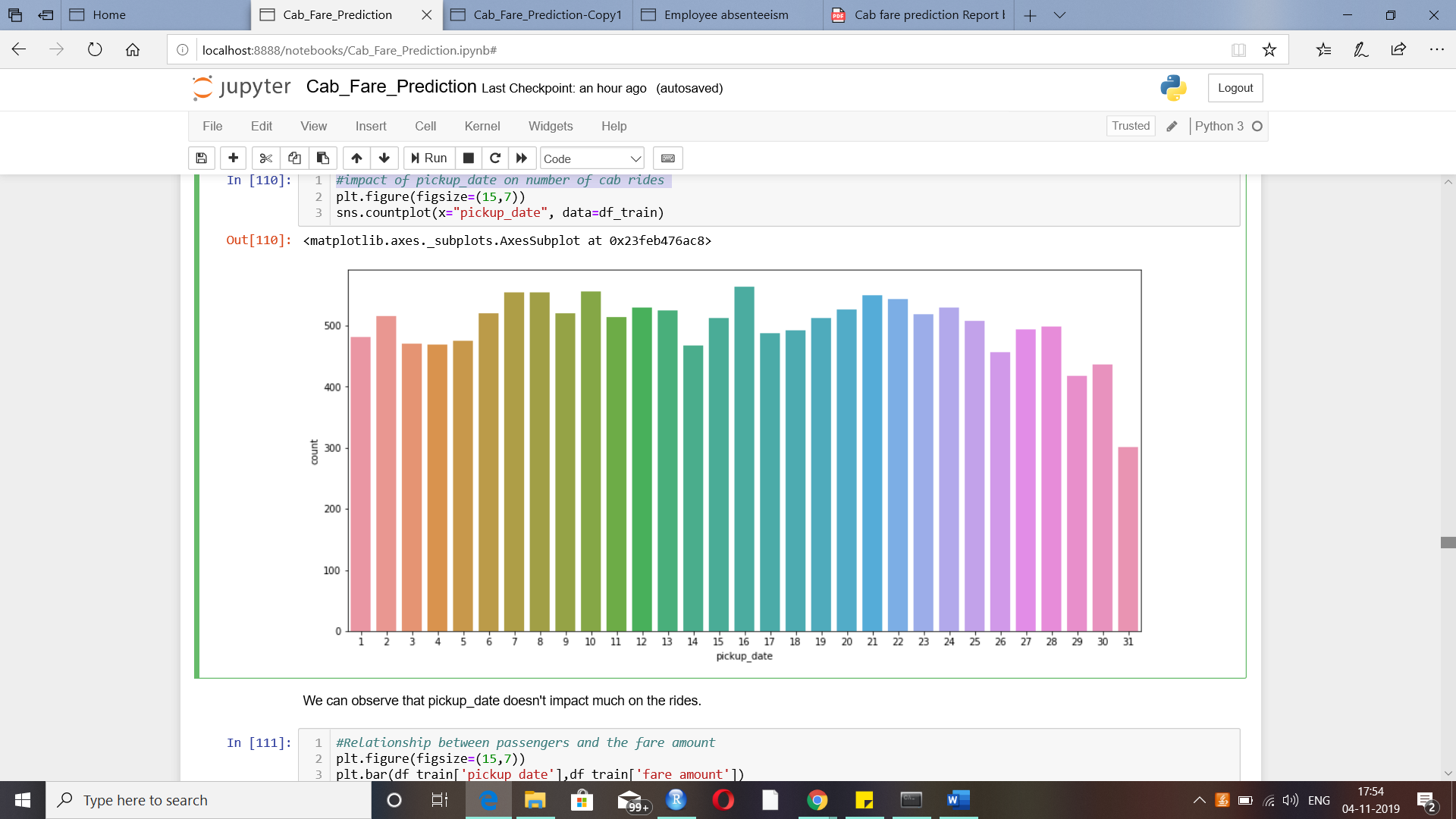
Data visualization helps us to understand the relationship between features. Here, to understand how each independent feature is related to the target feature, we need to perform visualization on data.

***1. Impact of passenger\_count on number of cab rides:***



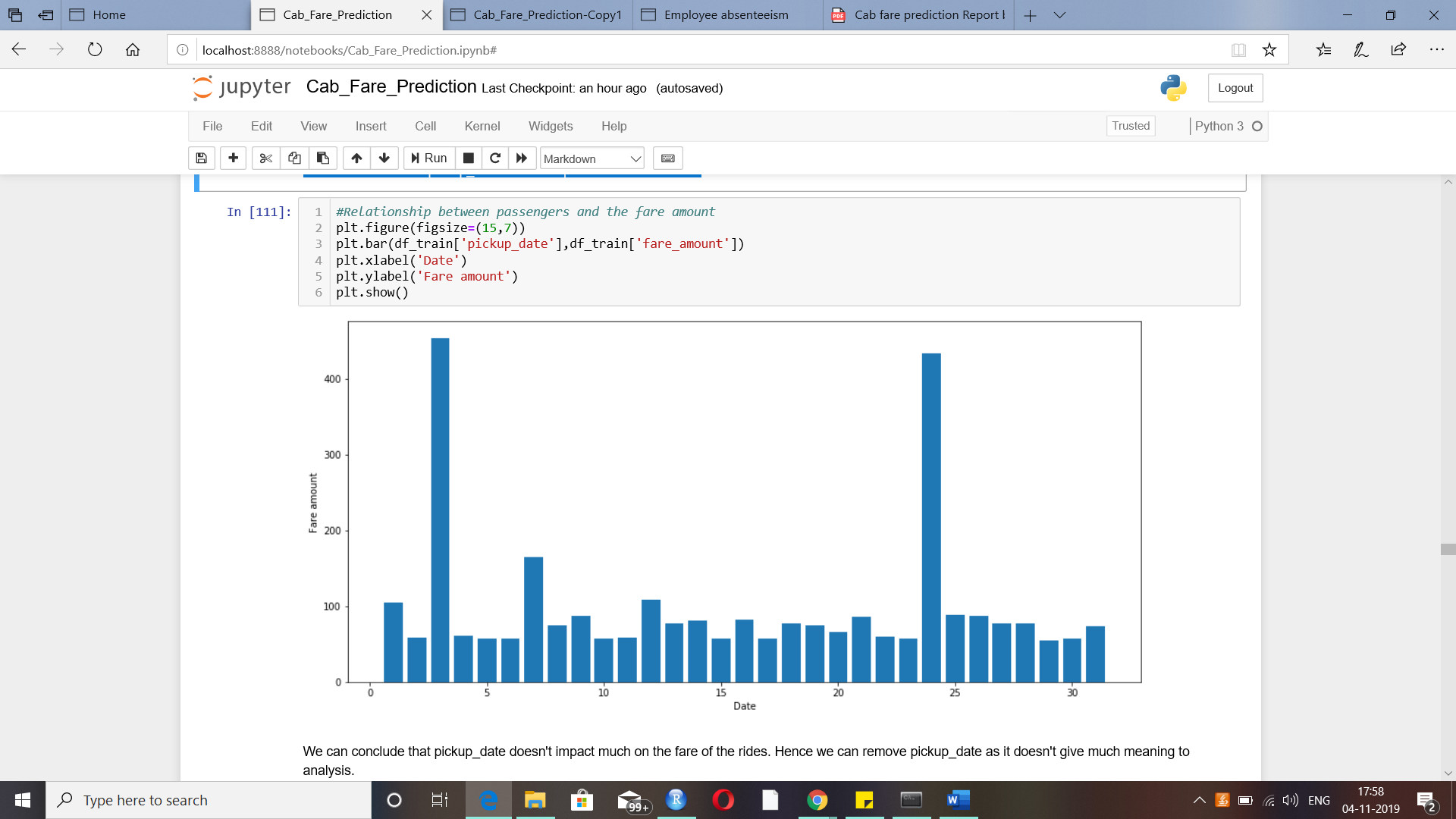
We can observe that 1 (single) passengers are the most frequent travellers.

***2. Impact of pickup\_date on number of cab rides:***



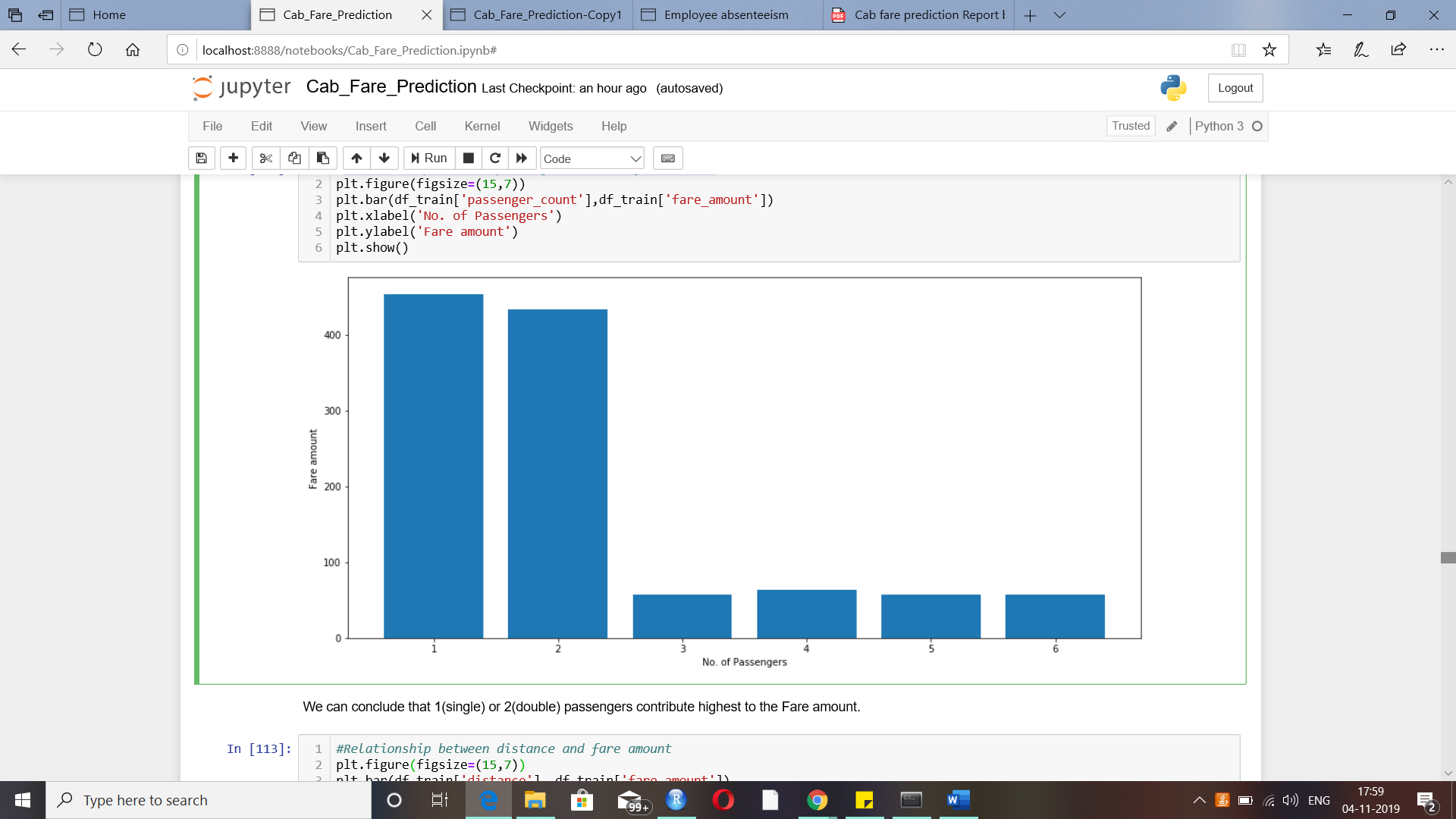
We can observe that pickup\_date doesn't impact much on the rides.

***3. Relationship between passengers and fare\_amount:***



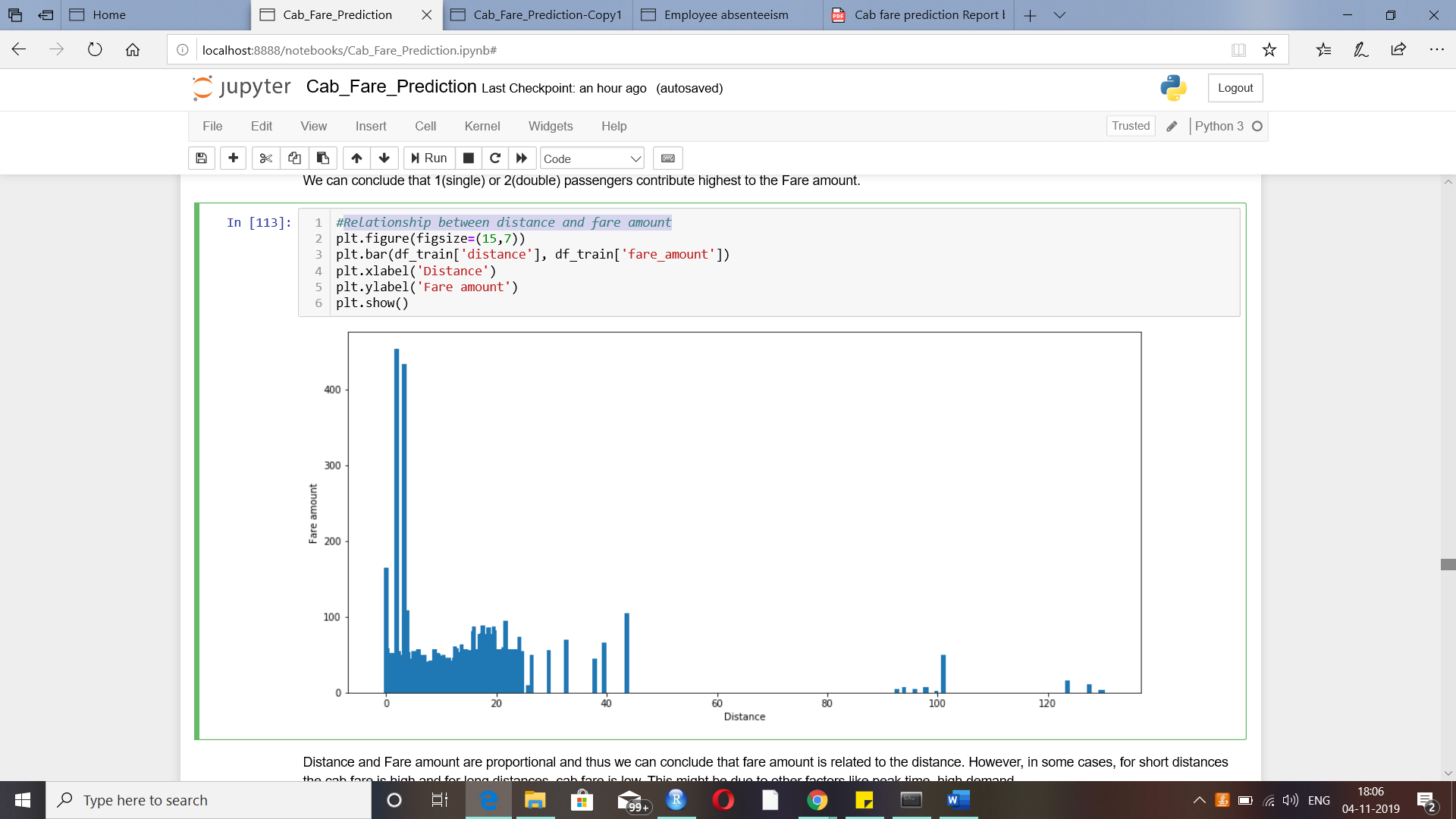
We can conclude that pickup\_date doesn't impact much on the fare of the rides. Hence we can remove pickup\_date as it doesn't give much meaning to analysis.

***4. Relationship between passengers and the fare amount:***



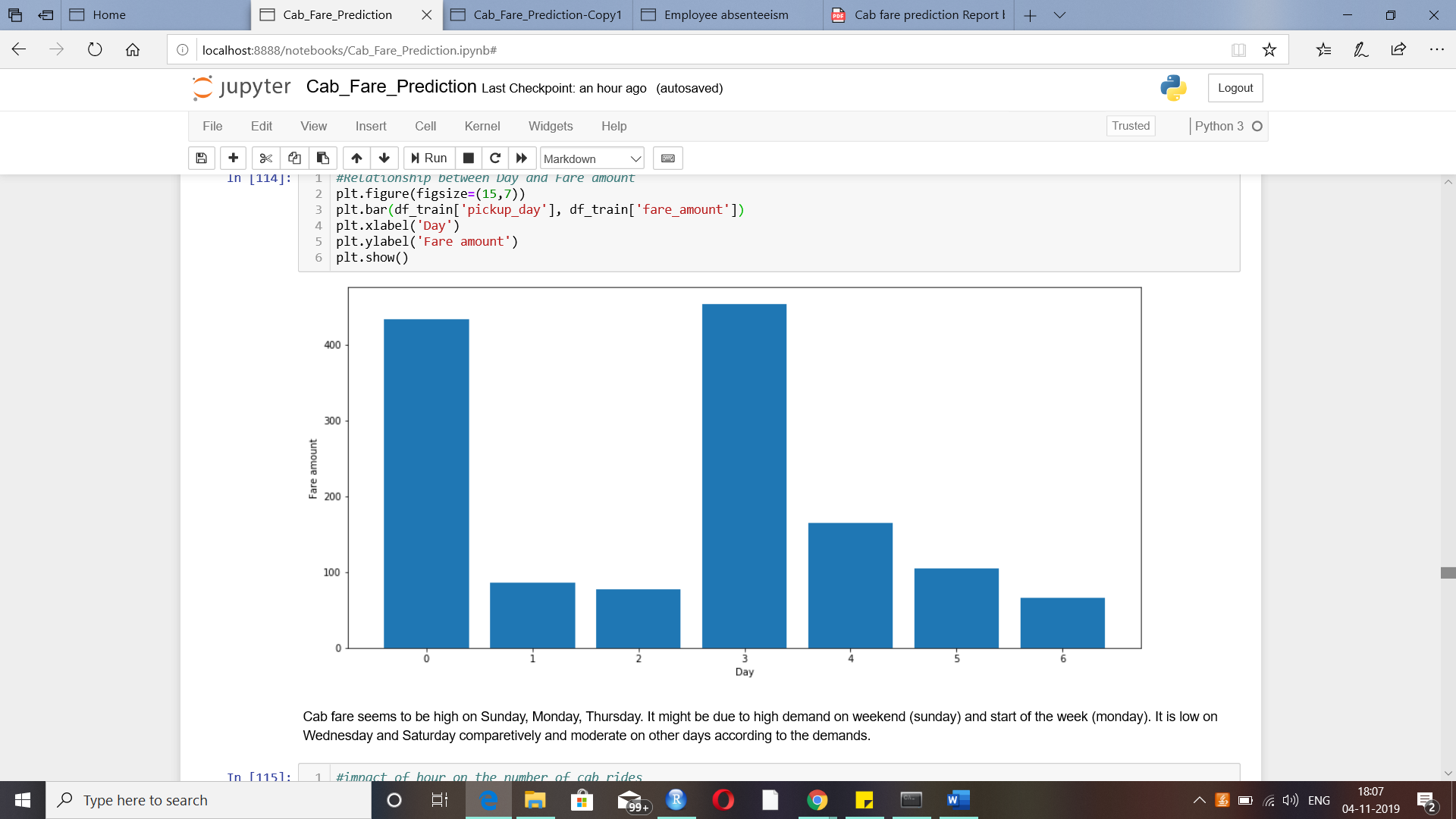
We can conclude that 1(single) or 2(double) passengers contribute highest to the Fare amount.

***5. Relationship between distance and fare amount:***



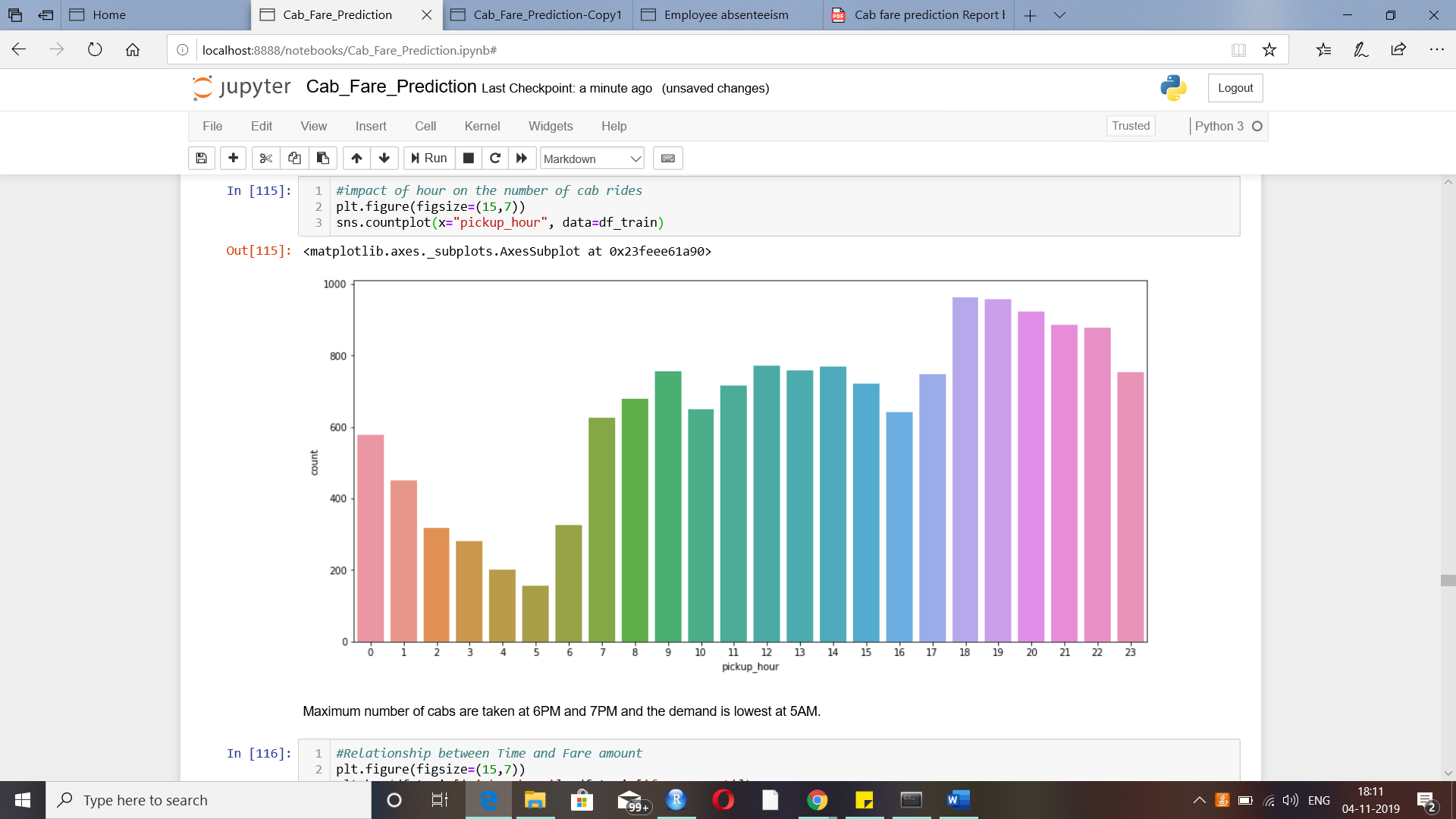
Distance and Fare amount are proportional and thus we can conclude that fare amount is related to the distance. However, in some cases, for short distances the cab fare is high and for long distances, cab fare is low. This might be due to other factors like peak time, high demand.

***6. Relationship between day and fare amount:***



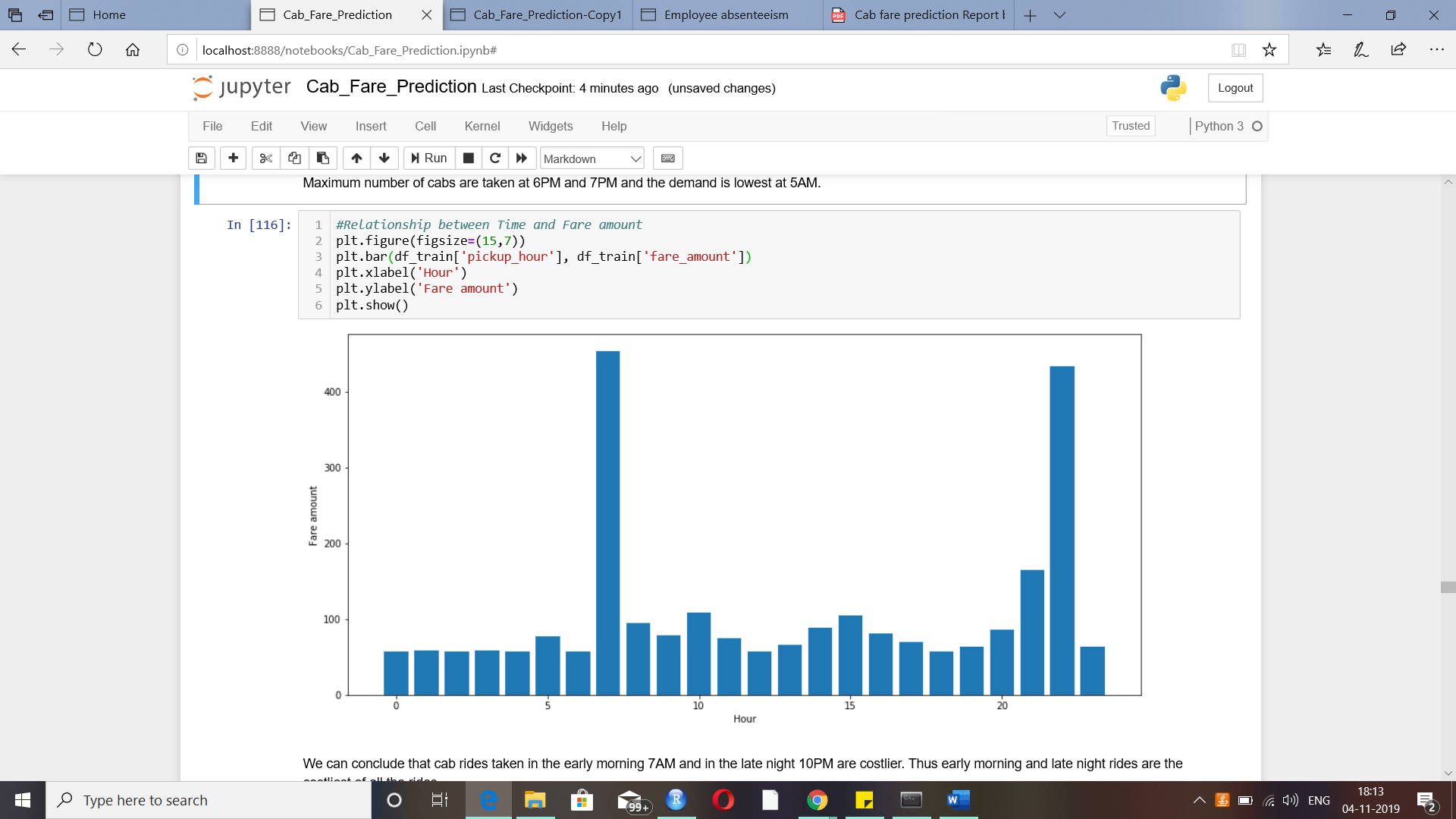
Cab fare seems to be high on Sunday, Wednesday, Thursday. It might be due to high demand on weekend (sunday) and middle of the week (wednesday). It is low on Tuesday and Saturday comparetively and moderate on other days according to the demands.

***7. Impact of the hour on the number of rides:***



Maximum number of cabs are taken at 6PM and 7PM and the demand is lowest at 5AM.

***8.Relationship between time and fare\_amount:***



We can conclude that cab rides taken in the early morning 7AM and in the late night 10PM are costlier. Thus, early morning and late night rides are the costliest of all the rides.

**Feature selection:**

Dependent variable:

-> fare\_amount

Independent variables:

* pickup\_latitude
* pickup\_longitude
* dropoff\_latitude
* dropoff\_longitude
* distance
* passenger\_count
* pickup\_year
* pickup\_date
* pickup\_month
* pickup\_day
* pickup\_hour

Correlation Matrix:

We can find the correlation between fare\_amount (target continuous variable) and independent continuous variables in training dataset using correlation matrix.

And if features are correlated with each other that could introduce bias into our models. Hence there should be no correlation between independent features and there should be high correlation between target and independent variables.

Observations:

1. We have derived different variables like year, month etc from pickup\_datetime variable and also we have calculated distance using pickup,dropoff latitude and longitude coordinates.
2. Also, from correlation matrix we can see that distance is highly correlated with fare\_amount(target variable) and pickup and dropoff latitudes and longitudes variables(dependent variables) are highly negatively correlated with each other. Hence, we can drop pickup\_datetime, pickup, dropoff latitude and longitude variables.

ANOVA test:

We can find the correlation between fare\_amount (target continuous variable) and independent categorical variables present in training dataset using ANOVA test of independence. It is compares the mean between each groups in a categorical variable.

Hypothesis of ANOVA testing:

Null Hypothesis: Mean of all categories in a variable are same and fare\_amount doesn't depend on it.

Alternate Hypothesis: Mean of at least one category in a variable is different and fare\_amount depends on it.

If p-value is less than 0.05 then we reject the null hypothesis. And if p-value is greater than 0.05 then we accept the null hypothesis.

Observations:

1. We can drop pickup\_date as it will of be much use to our analysis.

**Feature Scaling:**

As the continuous variables, distance and fare\_amount are varying in units and range,we can scale them before applying machine learning algorithms on them. We can perform normality check on variables to determine which method to be used to scale.

As the distribution of distance variable is right skewed, we can use log transformation to change the distribution.

Apply log transformation on fare\_amount and distance variables.

**3. Modelling:**

**3.1 Model Development:**

Once all the pre-processing has been done on the dataset, we will now be able to develop a model. The choice of machine learning algorithm depends upon the problem category. As per our problem statement we can develop following models:

• Linear regression

• Decision Tree

• Random forest

• Gradient Boosting

**Linear Regression:**

It is the simplest and powerful statistical model for prediction. It uses weights or coefficients of each independent variable in training data to develop model to be tested on test data.

Since, our training dataset has categorical and continuous variables, we have to encode values of categorical variables into numeric values and then pass as input.

**Decision tree:**

Decision tree is a supervised model for classification and Regression. It is used to create training model which is used to predict/classify values of target variable by learning decision rules based on historical data.

**Random Forest:**

Random Forest is an ensemble of decision trees where n number of random variables are used to construct n decision trees.

**Gradient Boosting:**

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.

**3.2 Model Evaluation:**

After having built a model, we have to measure how accurately the model predicts the response. RMSE (Root Mean Square Error)is one of the good measures and criterion for fit for prediction models.

The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model’s predicted values. As compared to mean absolute error, RMSE gives higher weightage and punishes large errors. The RMSE is directly interpretable in terms of measurement units, and so is a better measure of goodness of fit.

**3.3 Model Optimization:**

**Optimizing models using Hyperparameter tuning:**

Choosing the right set of hyperparameters values is “Hyperparameter optimization” or “Hyperparameter tuning”. The right choice of hyperparameter values affect the performance of a machine learning model.

Hyperparameter optimization/tuning strategies used are:

1. Grid Search
2. Random Search.

Grid Search is an approach to hyperparameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

In Random Search, a statistical distribution for each hyperparameter is provided from which values may be randomly sampled.

Here, we are using the below values for optimization:

n\_estimator = [200,500]

max\_depth = [2,3,4,5,6,7]

**3.4 Model Selection:**

After optimizing regression models, we can select the model which yields low RMSE value and high R-Squared value.

**4. Conclusion:**

Below are the train and test data results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | RMSE | | R-Squared | |
|  | Training dataset | Test dataset | Training dataset | Test dataset |
| Decision tree | 0.299 | 0.287 | 0.702 | 0.709 |
| Random Forest | 0.094 | 0.241 | 0.970 | 0.794 |
| Gradient Boosting | 0.228 | 0.230 | 0.826 | 0.813 |

R^2 values of test dataset are nearly same for Random Forest model and Gradient Boosting model, we can optimize them for better results.

Below table shows the model results after optimizing using hyper parameters tuning:

Model : Random Forest– GridSearchCV

R–Squared: 0.805

RMSE: 0.234

Model : Gradient Boosting – GridSearchCV

R–Squared: 0.811

RMSE: 0.230

Model :Random Forest – RandomSearchCV

R–Squared: 0.802

RMSE: 0.236

Model :Gradient Boosting – RandomSearchCV

R–Squared: 0.800

RMSE: 0.237

Hence, by optimizing Random Forest and Gradient Boosting models using GridSearchCV and RandomizedSearchCV algorithms, we can conclude that GridSearchCV for Gradient Boosting Model yields maximum R^2 value. We can predict test dataset using GridSearchCV for Gradient Boosting Model.

**Code in R:**

#Load libraries

library(caret) #for data splitting function - createDataPartition

library(rpart) #to build Decision tree regression model

library(randomForest) #to build random forest regression model

library(DMwR) #to calculate regression evaluation statistics

library(ggplot2) #for visualizations of data

library(gbm) #to build gradient boosting model

#set working directory

setwd("C:/Users/Usha/Edwisor/Project-Cab Fare Prediction")

#to check if the working directory is set right

getwd()

#Loading the training and test data set to model and predict values

df\_train = read.csv("train\_cab.csv", sep=",")

df\_test = read.csv("test.csv",sep=",")

#################################DATA EXPLORATION#############################################

##### Data Exploration or preparation includes,

##1. Identification of variables and their datatypes

##2. Descriptive statistics

##3. Conversion of data types into required ones

##4. Missing Value Analysis

##5. Outlier Analysis

##6. Feature Engineering

########### 1. Identification of variables and their datatypes############

#fetch first five observations from training dataset to identify target and predictor variables

head(df\_train)

#to fetch first five observations from test dataset

head(df\_test)

#to get the number of entries or dimensions of the training dataset

dim(df\_train)

#to get the dimensions of the test dataset

dim(df\_test)

#to get the data types of variables in training dataset

str(df\_train)

#to get the data types of variables in test dataset

str(df\_test)

#to get the unique values from training dataset

unique(df\_train)

################### 2. Descriptive statistics################

#to get the summary of training dataset

summary(df\_train)

#to get the list of unique values of passenger\_count

table(df\_train$passenger\_count)

#to get the unique values of fare\_amount

unique(df\_train$fare\_amount)

###\*\*Observations:

##1. There are rides consisted of even passengers greater than 500 and minimum as 0. These are clear outliers and points to data inconsistency.

##2. Most of the rides consists of passengers either 1 or 2.

##3. There are null values present in passenger\_count.

##4. There are 468 unique values of fare amount in the training data.

##5. There are observations in pickup\_latitude, dropoff\_latitude greater than 400 and less than -90.

##6. There are observations in pickup\_longitude, dropoff\_longitude less than -180.

#to get the distribution of test dataset

summary(df\_test)

###\*\*Observations:

##1. Passenger count consists of maximum value 6 and minimum value 1 which is within the range.

##2. Pickup and Dropoff latitudes and longitudes are not greater than 42 and -72 which is acceptable range for latitudes and longitudes.

################ 3. Conversion of datatypes into the required ones###############

# As observed, we have to change fare\_amount from factor to numeric

df\_train$fare\_amount = as.numeric(as.character(df\_train$fare\_amount))

summary(df\_train$fare\_amount)

##We will have to convert passenger\_count into a categorical variable because passenger\_count is not a continuous variable. passenger\_count cannot take continous values. and also they are limited in number if its a cab. It can be done after missing and outlier analysis are done.

##################### 4. Missing value Analysis#####################

#to get the sum of null/missing values in training set

sum(is.na(df\_train))

#create dataframe to analyze missing values in training dataset

missing\_val = data.frame(apply(df\_train,2,function(x){sum(is.na(x))}))

missing\_val$Columns = row.names(missing\_val)

names(missing\_val)[1] = "Missing\_count"

missing\_val$Missing\_percentage = (missing\_val$Missing\_count/nrow(df\_train)) \* 100

missing\_val = missing\_val[order(-missing\_val$Missing\_percentage),]

row.names(missing\_val) = NULL

missing\_val = missing\_val[,c(2,1,3)]

missing\_val

#to get the sum of null/missing values in test set

sum(is.na(df\_test))

###\*\*Observations:

#1. Since the percentage of missing values of passenger\_count and fare\_amount in training dataset is less, we can impute the missing values statistically.

#2. However, there are no missing values present in test dataset.

#3. Imputation of missing values in training dataset:

# Imputation is a method to fill in the missing values with estimated ones. Mean / Mode / Median imputation is one of the most frequently used methods. It consists of replacing the missing data for a given attribute by the mean or median (quantitative attribute) or mode (qualitative attribute) of all known values of that variable.

# Since, fare\_amount and passenger\_count are numerical and categorical, we can opt for Mean/Median/Mode and KNN methods for imputation. To find the apt method out of all,

#1. Remove any random value from the column missing and replace with NA.

#2. Impute the null value with methods one by one.

#3. Select the method with output nearly equal to the original value.

## 1. Passenger count

unique(df\_train$passenger\_count)

df\_train$passenger\_count=round(df\_train$passenger\_count)

df\_train[,'passenger\_count'] = factor(df\_train[,'passenger\_count'], labels=(1:6))

df\_test[,'passenger\_count'] = factor(df\_test[,'passenger\_count'], labels=(1:6))

#Choosing a random value from passenger\_count to replace it as NA

df\_train$passenger\_count[99]

#Replace 1.0 with NA

df\_train$passenger\_count[99] = NA

df\_train$passenger\_count[99]

df\_train$passenger\_count[99] = NA

#Impute with mode

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

# Mode Method

getmode(df\_train$passenger\_count)

df\_train$passenger\_count[99]

##Here, after imputation, the values are:

#Actual value - 1

#After Mode Imputation - 1

#After KNN Imputation - 1

#We can't use mode method as data will be more biased towards passenger\_count = 1 as it's proportion is high in dataset.

## 2. Fare amount

#Choosing a random fare\_amount value to replace it as NA

df\_train$fare\_amount[201]

#Replace 6 with NA

df\_train$fare\_amount[201] = NA

df\_train$fare\_amount[201]

#Impute with mean

df\_train$fare\_amount[201] = mean(df\_train$fare\_amount,na.rm=T)

df\_train$fare\_amount[201]

df\_train$fare\_amount[201] = NA

#Impute with median

df\_train$fare\_amount[201] = median(df\_train$fare\_amount,na.rm=T)

df\_train$fare\_amount[201]

df\_train$fare\_amount[201] = NA

###\*\*Observations:

##Here, after imputation, the values are:

#Actual value = 6.0

#After Mean Imputation - 15.01

#After Median Imputation - 8.5

#After KNN Imputation - 6.32

##We will go for KNN Imputations to impute both passenger\_count and fare\_amount missing values

#to get the observations with missing values in passenger\_count

df\_train[is.na(df\_train$passenger\_count),]

#to get the observations with missing values in fare\_amount

df\_train[is.na(df\_train$fare\_amount),]

#KNN Imputation

df\_train = knnImputation(df\_train, k = 181)

#to check for null values after removing missing value rows

sum(is.na(df\_train))

####################### 5. Outlier Analysis#####################

##\*\*Based on the basic understanding of the data from statistical analysis, outliers are the values that are not in the desired range. It is obvious that the outliers present in the training data are due to incorrectly entered or measured data. Hence, we can remove those values.

## 1. Pickup and Dropoff latitudes and longitudes

##\*\*Observation:

#pickup and dropoff latitude should be in the range of -90 to +90 and pickup and dropoff longitude should be in the range of -180 to +180. Hence, rows having pickup and dropoff latitude, longitude values out of the given range should be considered as outliers.

#fetch rows having pickup\_latitude less than -90

df\_train[which(df\_train$pickup\_latitude < -90),]

#fetch rows having pickup\_latitude greater than 90

df\_train[which(df\_train$pickup\_latitude > 90),]

#Drop the row whose pickup\_latitude > 90

df\_train = df\_train[-which(df\_train$pickup\_latitude > 90),]

#fetch rows having dropoff\_latitude less than -90

df\_train[which(df\_train$dropoff\_latitude < -90),]

#fetch rows having dropoff\_latitude greater than 90

df\_train[which(df\_train$dropoff\_latitude > 90),]

#fetch rows having pickup\_longitude less than -180

df\_train[which(df\_train$pickup\_longitude < -180),]

#fetch rows having pickup\_longitude greater than 180

df\_train[which(df\_train$pickup\_longitude > 180),]

#fetch rows having dropoff\_longitude less than -180

df\_train[which(df\_train$dropoff\_longitude < -180),]

#fetch rows having dropoff\_longitude greater than 180

df\_train[which(df\_train$dropoff\_longitude > 180),]

###\*\*Observation:

#Thus, pickup, dropoff latitude and longitude variables are within the range in training dataset and there are no outliers in pickup, dropoff latitude and longitude variables in test dataset.

## 2. Passenger count

###\*\*Observations:

#1.Maximum number of passengers that can travel in a car can be only 6. Hence, values greater than 6 are outliers.

#2.Passenger count cannot be 0. Hence, any value less than or equal to zero is an outlier.

#get the rows whose passenger\_count values are greater than 6

df\_train[which(df\_train$passenger\_count > 6),]

#remove rows whose passenger\_count is greater than 6

df\_train = df\_train[-which(df\_train$passenger\_count > 6),]

#get the rows whose passenger\_count values are less than 0

df\_train[which(df\_train$passenger\_count < 0),]

#get the rows whose passenger\_count values are 0

df\_train[which(df\_train$passenger\_count == 0),]

#remove rows whose passenger\_count is equal to 0

df\_train = df\_train[-which(df\_train$passenger\_count == 0),]

#arrange passenger\_count in ascending order to get outliers from passenger\_count

print(df\_train[order(df\_train$passenger\_count,decreasing = FALSE),])

##Now, maximum and minimum values of passenger\_count are within the range.

## 3.Fare amount

summary(df\_train$fare\_amount)

##\*\*Observation:

#Fare amount can't be zero or less than that. Hence, values less than or equal to 0 are outliers.

#to get rows having fare\_amount less than 0

df\_train[which(df\_train$fare\_amount < 0),]

#remove rows with fare\_amount value less than 0

df\_train = df\_train[-which(df\_train$fare\_amount < 0),]

#to get rows having fare\_amount equal to 0

df\_train[which(df\_train$fare\_amount == 0),]

#remove rows with fare\_amount value equal to 0

df\_train = df\_train[-which(df\_train$fare\_amount == 0),]

#arrange fare\_amount in descending order to get outliers from fare\_amount

print(df\_train[order(df\_train$fare\_amount,decreasing = TRUE),])

##\*\*Observation:

#Now, there is a huge difference in the fare amounts of first two observations and the rest of the dataset. We can remove them as they are outliers.

#remove rows with fare\_amount value greater than 454

df\_train = df\_train[-which(df\_train$fare\_amount >454),]

#fetch the minimum value of fare\_amount

min(df\_train$fare\_amount)

#remove fare\_amount minimum value 0.01 as value can never be this low

df\_train = df\_train[-which(df\_train$fare\_amount == 0.01),]

###\*\*Observations:

#1. Now, maximum and minimum values of fare amount are in the range.

#2. Hence, there are no outliers present in fare\_amount in training dataset. There are no missing values and outliers present in test dataset.

####################### 5. Feature Engineering####################

############Feature Creation################

## 1.Distance

##We have latitude and longitude coordinates. Thus we can calculate distance between the coordinates.This can be done based on haversine formula. The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Here, we create a new derived feature - distance.

deg\_to\_rad = function(deg){

(deg \* pi) / 180

}

haversine = function(long1,lat1,long2,lat2){

#long1rad = deg\_to\_rad(long1)

phi1 = deg\_to\_rad(lat1)

#long2rad = deg\_to\_rad(long2)

phi2 = deg\_to\_rad(lat2)

delphi = deg\_to\_rad(lat2 - lat1)

dellamda = deg\_to\_rad(long2 - long1)

a = sin(delphi/2) \* sin(delphi/2) + cos(phi1) \* cos(phi2) \*

sin(dellamda/2) \* sin(dellamda/2)

c = 2 \* atan2(sqrt(a),sqrt(1-a))

R = 6371e3

R \* c / 1000 #1000 is used to convert to meters

}

#apply haversine function to training dataset

df\_train$distance = haversine(df\_train$pickup\_longitude,df\_train$pickup\_latitude,df\_train$dropoff\_longitude,df\_train$dropoff\_latitude)

head(df\_train)

#apply haversine function to test dataset

df\_test$distance = haversine(df\_test$pickup\_longitude,df\_test$pickup\_latitude,df\_test$dropoff\_longitude,df\_test$dropoff\_latitude)

head(df\_test)

#to get the distribution of distance in training dataset

summary(df\_train$distance)

#to get the distribution of distance in test dataset

summary(df\_test$distance)

#arrange distance values in descending order to find whether outliers are present or not

print(df\_train[order(df\_train$distance,decreasing = TRUE),])

###\*\*Observations:

##We get to know that there is a drop in the value of distance from 4447.08 to 129.95 and there is no increase after it. Hence,

#1. Distance values greater than 130 could be considered as outliers and removed.

#2. Distance cannot be zero. The rows containing distance equal to 0 can also be dropped.

#drop rows whose distance value is equal to zero

df\_train = df\_train[-which(df\_train$distance == 0),]

#drop rows whose distance value is greater than 130 (rounding off 129.95 max value)

df\_train = df\_train[-which(df\_train$distance > 130),]

#arrange distance values in descending order to ensure there are no outliers

print(df\_train[order(df\_train$distance,decreasing = TRUE),])

#arrange distance values in descending order to check for oultiers in test dataset

print(df\_test[order(df\_test$distance,decreasing = TRUE),])

#to get the count of distance value equal to 0 in test dataset

df\_test[which(df\_test$distance == 0),]

#drop rows whose distance value is equal to 0

df\_test = df\_test[-which(df\_test$distance == 0),]

print(df\_test[order(df\_test$distance,decreasing = TRUE),])

###\*\*Observation:

#Thus, there are no outliers in distance variable in both training and test datasets.

## 2. Year, Month, Day, Date, Hour

## Now, we have pickup\_datetime feature with us. We can generate new variables like year, month, date, day, hour from it which might have better relationship with our target variable. This also helps to highlight the hidden relationship in a variable.

df\_train$pickup\_date = as.Date(as.character(df\_train$pickup\_datetime))

df\_train$pickup\_day = as.factor(format(df\_train$pickup\_date,"%u"))# Monday = 1

df\_train$pickup\_month = as.factor(format(df\_train$pickup\_date,"%m"))

df\_train$pickup\_year = as.factor(format(df\_train$pickup\_date,"%Y"))

pickup\_time = strptime(df\_train$pickup\_datetime,"%Y-%m-%d %H:%M:%S")

df\_train$pickup\_hour = as.factor(format(pickup\_time,"%H"))

df\_train$passenger\_count = as.factor(df\_train$passenger\_count)

summary(df\_train)

#to get the observation which contains 'pickup\_date' as <NA>

df\_train[is.na(df\_train['pickup\_date']),]

####We have one <NA> value induced in pickup\_date and pickup\_datetime value is 43. We can remove the observation as it is an outlier and may be incorrectly entered data.

#Drop the observation that has value of 'pickup\_date' as <NA>

df\_train = df\_train[-which(is.na(df\_train['pickup\_date'])),]

## Deriving new variables from pickup\_datetime in test dataset.

df\_test$pickup\_date = as.Date(as.character(df\_test$pickup\_datetime))

df\_test$pickup\_day = as.factor(format(df\_test$pickup\_date,"%u"))# Monday = 1

df\_test$pickup\_month = as.factor(format(df\_test$pickup\_date,"%m"))

df\_test$pickup\_year = as.factor(format(df\_test$pickup\_date,"%Y"))

pickup\_time = strptime(df\_test$pickup\_datetime,"%Y-%m-%d %H:%M:%S")

df\_test$pickup\_hour = as.factor(format(pickup\_time,"%H"))

df\_test$passenger\_count = as.factor(df\_test$passenger\_count)

summary(df\_test)

############# Feature Selection ######################

##Correlation Matrix:

#We can find the correlation between fare\_amount (target continuous variable) and independent continuous variables in training dataset using correlation matrix.

#And if features are correlated with each other that could introduce bias into our models. Hence there should be no correlation between independent features and there should be high correlation between target and independent variables.

numeric\_index = sapply(df\_train,is.numeric) #selecting only numeric

numeric\_data = df\_train[,numeric\_index]

#to find the correlation between variables in training dataset

corrgram::corrgram(df\_train[,numeric\_index],upper.panel=corrgram::panel.pie, main = "Correlation Plot")

###\*\*Observations:

#1. Now we have derived different variables like year, month etc from pickup\_datetime variable and also we have calculated distance using pickup,dropoff latitude and longitude coordinates.

#2. Also, from correlation matrix we can see that distance is highly correlated with fare\_amount(target variable) and pickup and dropoff latitudes and longitudes variables(dependent variables) are highly negatively correlated with each other. Hence, we can drop pickup\_datetime, pickup, dropoff latitude and longitude variables.

##ANOVA test:

#We an find the correlation between fare\_amount (target continuous variable) and independent categorical variables present in training dataset using ANOVA test of independence. It is compares the mean between each groups in a categorical variable.

#Hypothesis of ANOVA testing :

#Null Hypothesis: Mean of all categories in a variable are same and fare\_amount doesn't depend on it.

#Alternate Hypothesis: Mean of at least one category in a variable is different and fare\_amount depends on it.

#If p-value is less than 0.05 then we reject the null hypothesis.

#And if p-value is greater than 0.05 then we accept the null hypothesis.

aov\_results = aov(fare\_amount ~ passenger\_count + pickup\_hour + pickup\_month + pickup\_year+ pickup\_date + pickup\_day,data = df\_train)

summary(aov\_results)

#We can reject null hypothesis as fare\_amount is dependent on the pickup\_year,pickup\_month and pickup\_hour.

##Dropping above mentioned variables from training dataset after analyzing all the variables

df\_train = subset(df\_train,select = -c(pickup\_longitude,pickup\_latitude,dropoff\_longitude,dropoff\_latitude))

df\_test = subset(df\_test,select = -c(pickup\_longitude,pickup\_latitude,dropoff\_longitude,dropoff\_latitude))

df\_train = subset(df\_train,select=-c(pickup\_date,pickup\_datetime,pickup\_day,passenger\_count))

df\_test = subset(df\_test,select=-c(pickup\_date,pickup\_datetime,pickup\_day,passenger\_count))

############# Feature Scaling #########################

##As the continuous variables, distance and fare\_amount are varying in units and range,we can scale them before applying machine learning algorithms on them. We can perform normality check on variables to determine which method to be used to scale.

#plot histogram to check for distribution of distance variable in training dataset

ggplot2::ggplot(df\_train, ggplot2::aes\_string(x = df\_train$distance)) +

ggplot2::geom\_histogram(fill="blue", colour = "black") + ggplot2::geom\_density() +

ggplot2::theme\_bw() + ggplot2::xlab("distance") + ggplot2::ylab("Frequency")+ggplot2::ggtitle(" distribution of distance")

## As the distribution of distance variable is right skewed, we can use log transformation to change the distribution.

# Lets define function for log transformation of variables

signedlog10 = function(x) {

ifelse(abs(x) <= 1, 0, sign(x)\*log10(abs(x)))

}

# Applying log function to distance variable

df\_train$distance = signedlog10(df\_train$distance)

#check distribution after applying log transform method

ggplot2::ggplot(df\_train, ggplot2::aes\_string(x = df\_train$distance)) +

ggplot2::geom\_histogram(fill="blue", colour = "black") + ggplot2::geom\_density() +

ggplot2::theme\_bw() + ggplot2::xlab("distance") + ggplot2::ylab("Frequency")+ggplot2::ggtitle(" distribution of distance")

#plot histogram to check for distribution of fare\_amount variable in training dataset

ggplot2::ggplot(df\_train, ggplot2::aes\_string(x = df\_train$fare\_amount)) +

ggplot2::geom\_histogram(fill="blue", colour = "black") + ggplot2::geom\_density() +

ggplot2::theme\_bw() + ggplot2::xlab("fare amount") + ggplot2::ylab("Frequency")+ggplot2::ggtitle(" distribution of distance")

## As the distribution of fare amount variable is right skewed, we can use log transformation to change the distribution.

# Lets define function for log transformation of variables

signedlog10 = function(x) {

ifelse(abs(x) <= 1, 0, sign(x)\*log10(abs(x)))

}

# Applying log function to distance variable

df\_train$distance = signedlog10(df\_train$fare\_amount)

#check distribution after applying log transform method

ggplot2::ggplot(df\_train, ggplot2::aes\_string(x = df\_train$fare\_amount)) +

ggplot2::geom\_histogram(fill="blue", colour = "black") + ggplot2::geom\_density() +

ggplot2::theme\_bw() + ggplot2::xlab("fare amount") + ggplot2::ylab("Frequency")+ggplot2::ggtitle(" distribution of distance")

#################### Applying Machine Learning algorithms ###########################

## We can split training data into train and test datasets. Training dataset is used for building training model and test dataset is for validating our model. This is done to understand the robustness, accuracy and performance of the model built.

#Obtain test and training dataset

train.index = caret::createDataPartition(df\_train$fare\_amount, p = .80, list = FALSE)

train\_data = df\_train[train.index,]

test\_data = df\_train[-train.index,]

################## 1. Linear regression ##################

lm\_model = lm(fare\_amount~.,train\_data)

summary(lm\_model)

str(test\_data)

test\_data[,2:5]

predictions\_LM = predict(lm\_model, test\_data[,2:5])

ggplot2::qplot(x = test\_data[,1], y = predictions\_LM, data = test\_data, color = I("blue"), geom = "point")

RMSE = function(act, pred){

sqrt(mean((act - pred)^2))

}

RMSE(test\_data[,1],predictions\_LM)

################## 2. Decision Tree ##################

Dt\_model = rpart(fare\_amount~ ., data = train\_data, method = "anova")

summary(Dt\_model)

predictions\_DT = predict(Dt\_model, test\_data[,2:5])

RMSE = function(act, pred){

sqrt(mean((act - pred)^2))

}

RMSE(test\_data[,1],predictions\_DT)

################## 3. Random Forest ###################

rf\_model = randomForest(fare\_amount ~.,data=train\_data, importance = TRUE, ntree = 200)

summary(rf\_model)

predictions\_RF = predict(rf\_model, test\_data[,2:5])

RMSE = function(act, pred){

sqrt(mean((act - pred)^2))

}

regr.eval(test\_data[,1],predictions\_RF)

################## 4. Gradient Boosting ###################

gbm\_model = gbm(fare\_amount~.,data = train\_data, n.trees = 200, shrinkage = 0.01,interaction.depth = 5)

summary(gbm\_model)

predictions\_GBM = predict(gbm\_model, test\_data[,2:5],n.trees = 10000)

regr.eval(test\_data[,1],predictions\_GBM)

########PREDICTION OF FARE AMOUNT FOR TEST DATASET##################################

##Thus,we can build prediction model for test dataset using Gradient boosting method.

gbm\_model = gbm(fare\_amount~.,data = train\_data , n.trees = 10000, shrinkage = 0.01,interaction.depth = 5)

summary(gbm\_model)

predictions\_GBM = predict(gbm\_model, df\_test ,n.trees = 10000)

df\_test$Predicted\_fare\_amount = predictions\_GBM