Cab Fare Prediction

Vinod Pawar

20/10/2019

Contents

1 Introduction

1.1 Problem Statement 2

1.2 Data 2

2 Methodology

2.1 Pre-Processing 3

2.2 Removing values which are not 5

2.3 Missing Value Analysis 5

2.4 Outlier Analysis 8

2.5 Feature Engineering 9

2.6 Feature Selection 11

2.7 Feature Scaling 12

2.8 Splitting train and test Dataset 13

3 Model Development 13

4 r- Code

Introduction

1.1 Problem Statement

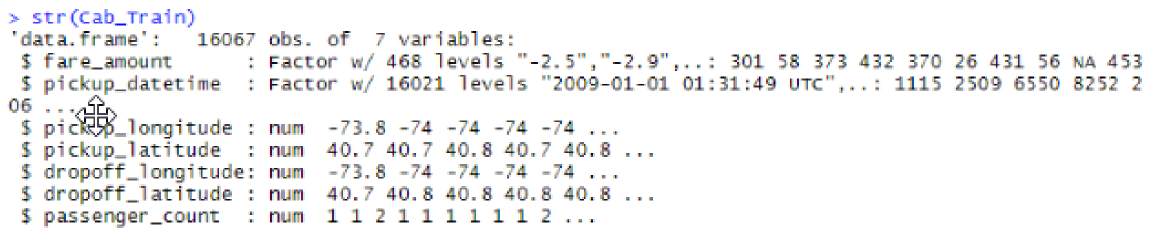
The objective of this project is to predict Cab Fare amount.

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 :- Given Below is sample dataset and an overview of observation

Attributes: ·

- pickup_datetime - timestamp value indicating when the cab ride started.
- pickup_longitude - float for longitude coordinate of where the cab ride started.
- pickup_latitude - float for latitude coordinate of where the cab ride started.
- dropoff_longitude - float for longitude coordinate of where the cab ride ended.
- dropoff_latitude - float for latitude coordinate of where the cab ride ended.
- passenger_count - an integer indicating the number of passengers in the cab ride.



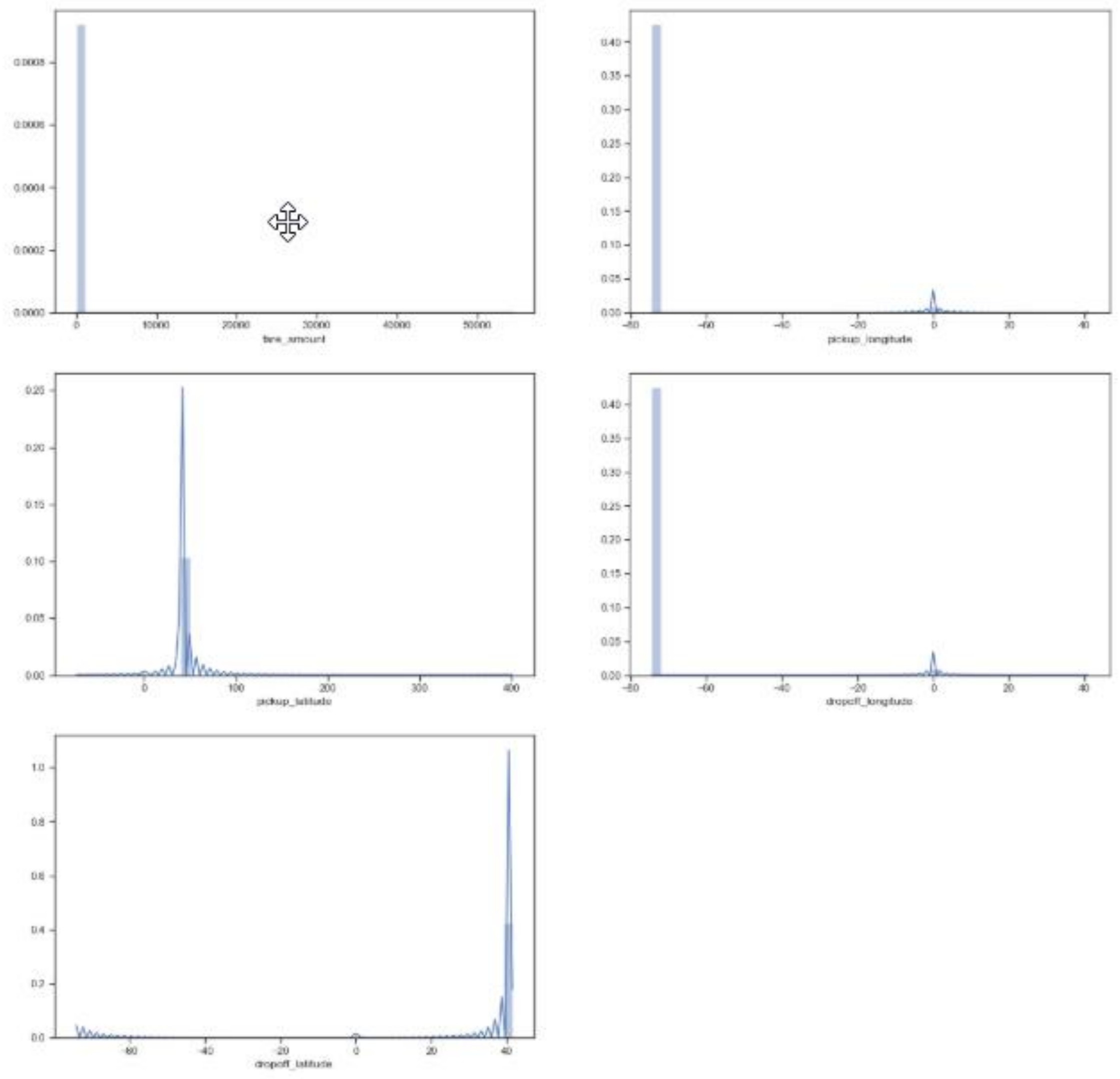
```
> str(Cab_Train)
'data.frame':	16067 obs. of  7 variables:
 $ fare_amount      : Factor w/ 468 levels "-2.5","-2.9",..: 301 58 373 432 370 26 431 56 NA 453
 $ pickup_datetime  : Factor w/ 16021 levels "2009-01-01 01:31:49 UTC",..: 1115 2509 6550 8252 2
06 ...
 $ pickup_longitude : num  -73.8 -74 -74 -74 -74 ...
 $ pickup_latitude  : num  40.7 40.7 40.8 40.7 40.8 ...
 $ dropoff_longitude: num  -73.8 -74 -74 -74 -74 ...
 $ dropoff_latitude : num  40.7 40.8 40.8 40.8 40.8 ...
 $ passenger_count  : num  1 1 2 1 1 1 1 1 1 2 ...
```

Methodology

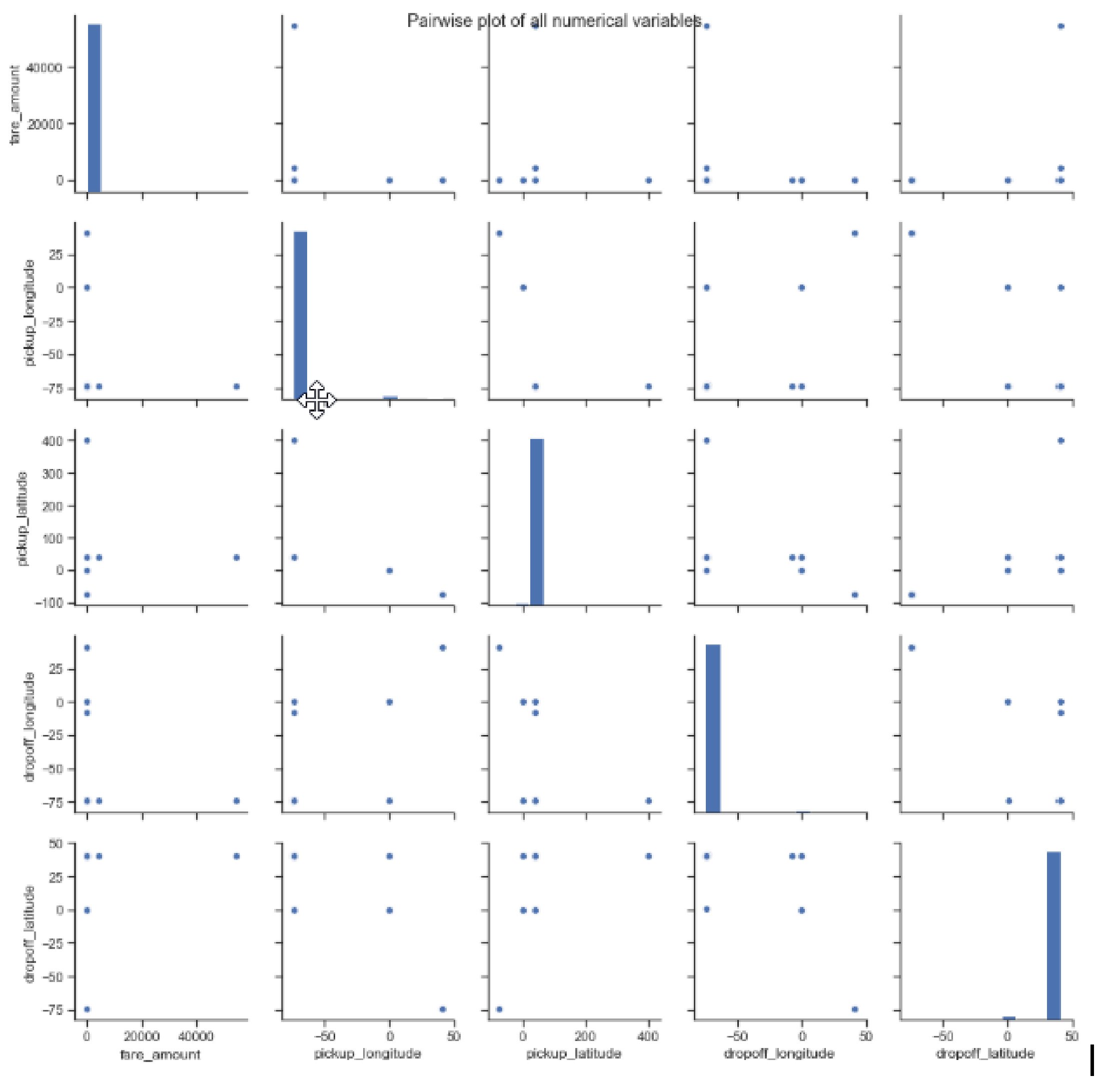
2.1 Pre-Processing

Data pre-processing is the first stage of any type of project. In this stage we get the feel of the data. We do this by looking at plots of independent variables vs target variables. If the data is messy, we try to improve it by sorting deleting extra rows and columns. This stage is called as Exploratory Data Analysis. This stage generally involves data cleaning, merging, sorting, looking for outlier analysis, looking for missing values in the data, imputing missing values if found by various methods such as mean, median, mode, KNN imputation, etc.
Further we will look into what Pre-Processing steps do this project was involved in.
Getting feel of data via visualization:

Some Histogram plots from seaborn library for each individual variable created.



Pairwise Plots for all Numerical variables:



2.2 Removing erroneous/invalid values.

In this step we will remove values in each variable which are not within desired range and we will consider them as outliers depending upon basic understanding of all the variables.

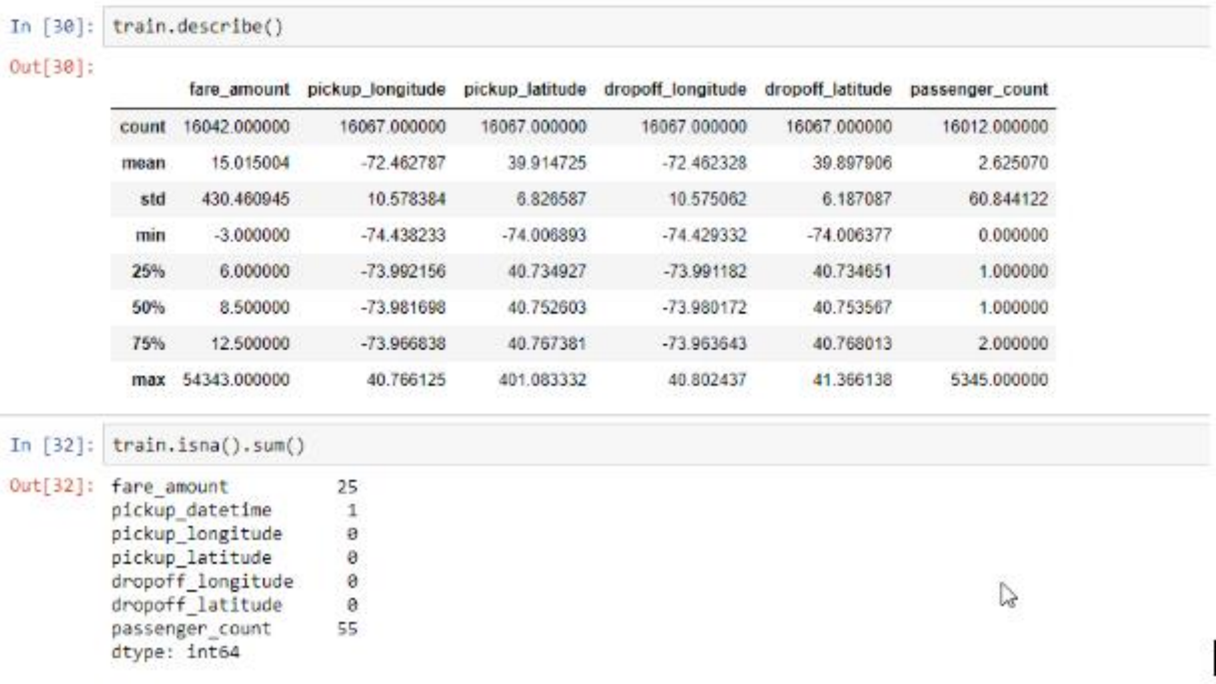
Example: - We will remove the passenger count 54236 of an observation, as this scenario is invalid there cannot be a passenger where this much number could accommodate in a cab.

Similarly we removed observations:-

1. Passenger count greater than 6
2. Passenger count less than 1 or equal to 0
3. Fare amount greater than 480
4. Latitudes range from -90 to 90.Longitudes range from -180 to 180.Removing which does not satisfy these ranges

2.3 Missing value Analysis

In this step we look for missing values in the dataset like empty row column cell which was left after removing special characters and punctuation marks. Some missing values are in form of NA. missing values left behind after outlier analysis; missing values can be in any form. Unfortunately, in this dataset we have found some missing values. Therefore, we will do some missing value analysis. Before imputed we selected random row no-1000 and made it NA, so that we will compare original value with imputed value and choose best method which will impute value closer to actual value.



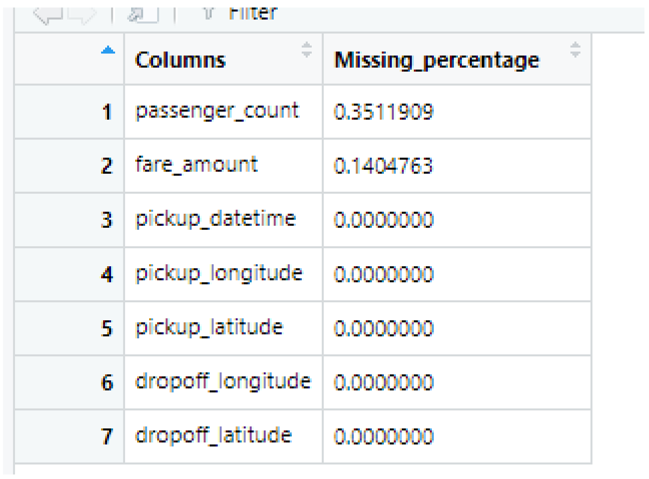
```
In [30]: train.describe()
Out[30]:
```

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	16042.000000	16067.000000	16067.000000	16067.000000	16067.000000	16012.000000
mean	15.015004	-72.462787	39.914725	-72.462328	39.897906	2.625070
std	430.460945	10.578384	6.826587	10.575062	6.187087	60.844122
min	-3.000000	-74.438233	-74.006893	-74.429332	-74.006377	0.000000
25%	6.000000	-73.992156	40.734927	-73.991182	40.734651	1.000000
50%	8.500000	-73.981698	40.752603	-73.980172	40.753567	1.000000
75%	12.500000	-73.966838	40.767381	-73.963643	40.768013	2.000000
max	54343.000000	40.766125	401.083332	40.802437	41.366138	5345.000000

```
In [32]: train.isna().sum()
Out[32]: fare_amount          25
         pickup_datetime       1
         pickup_longitude      0
         pickup_latitude       0
         dropoff_longitude     0
         dropoff_latitude      0
         passenger_count      55
         dtype: int64
```

We will impute values for fare_amount and passenger_count both of them has missing values 22 and 55 respectively. We will drop 1 value in pickup_datetime i.e it will be an entire row to drop.

Below are the missing value percentages for each variable



	Columns	Missing_percentage
1	passenger_count	0.3511909
2	fare_amount	0.1404763
3	pickup_datetime	0.0000000
4	pickup_longitude	0.0000000
5	pickup_latitude	0.0000000
6	dropoff_longitude	0.0000000
7	dropoff_latitude	0.0000000

We'd tried central statistical methods and algorithmic method--KNN to impute missing values in the dataset:

1. **For Passenger_coun**
   Actual value = 1
   Mode = 1
   KNN = 1

We will choose the KNN method here because it maintains the standard deviation of variable. We will not use Mode method because whole variable will be more biased towards 1 passenger_count also passenger_count has maximum value equals to 1

2. **For fare_amount**:

   Actual value =10.5,
   Mean = 11.366,
   Median = 8.5,
   KNN = 10.833

We will Choose KNN method here because it imputes value closest to actual value.

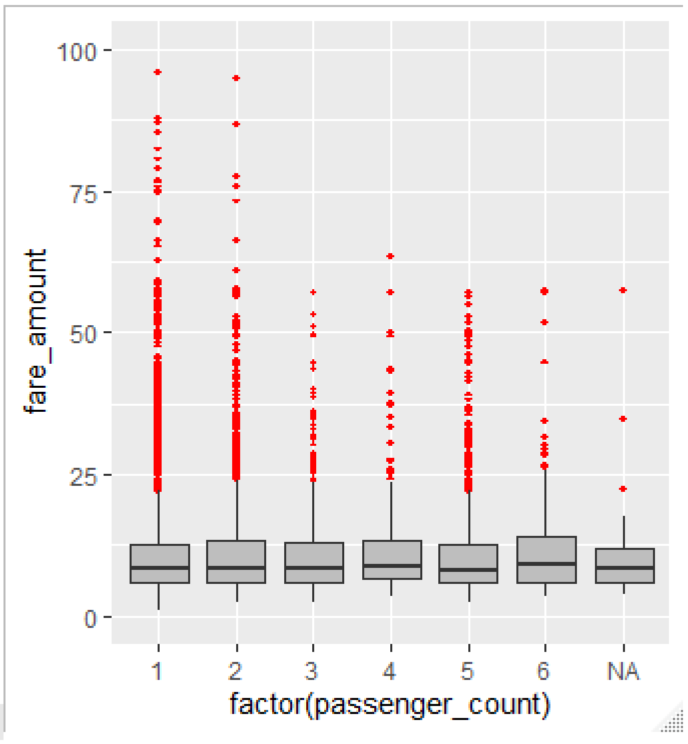
2.4 Outlier Analysis

We look for outlier in the dataset by plotting Boxplots. There are outliers present in the data. we have removed these outliers. This is how we done,

I. We replaced them with Nan values or we can say created missing values.

II. Then we imputed those missing values with KNN method.

- We Will do Outlier Analysis only on Fare_amount just for now and we will do outlier analysis after feature engineering latitudes and longitudes.



From above Boxplots we see that 'fare_amount'have outliers in it: 'fare_amount' has 1359 outliers.

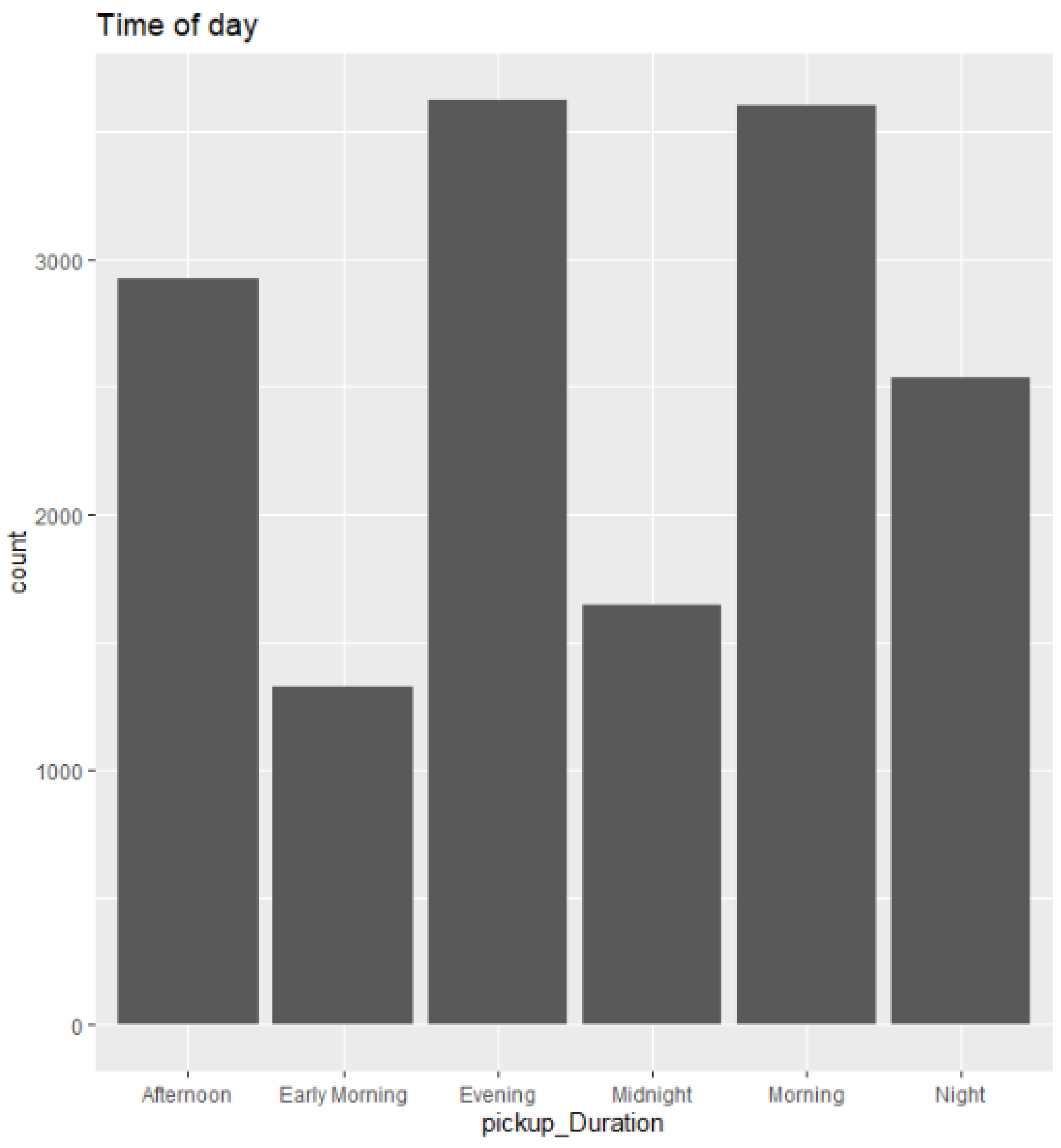
We successfully imputed these outliers with KNN and K value is 3

2.5 Feature Engineering

Feature Engineering is used to drive new features from existing features.

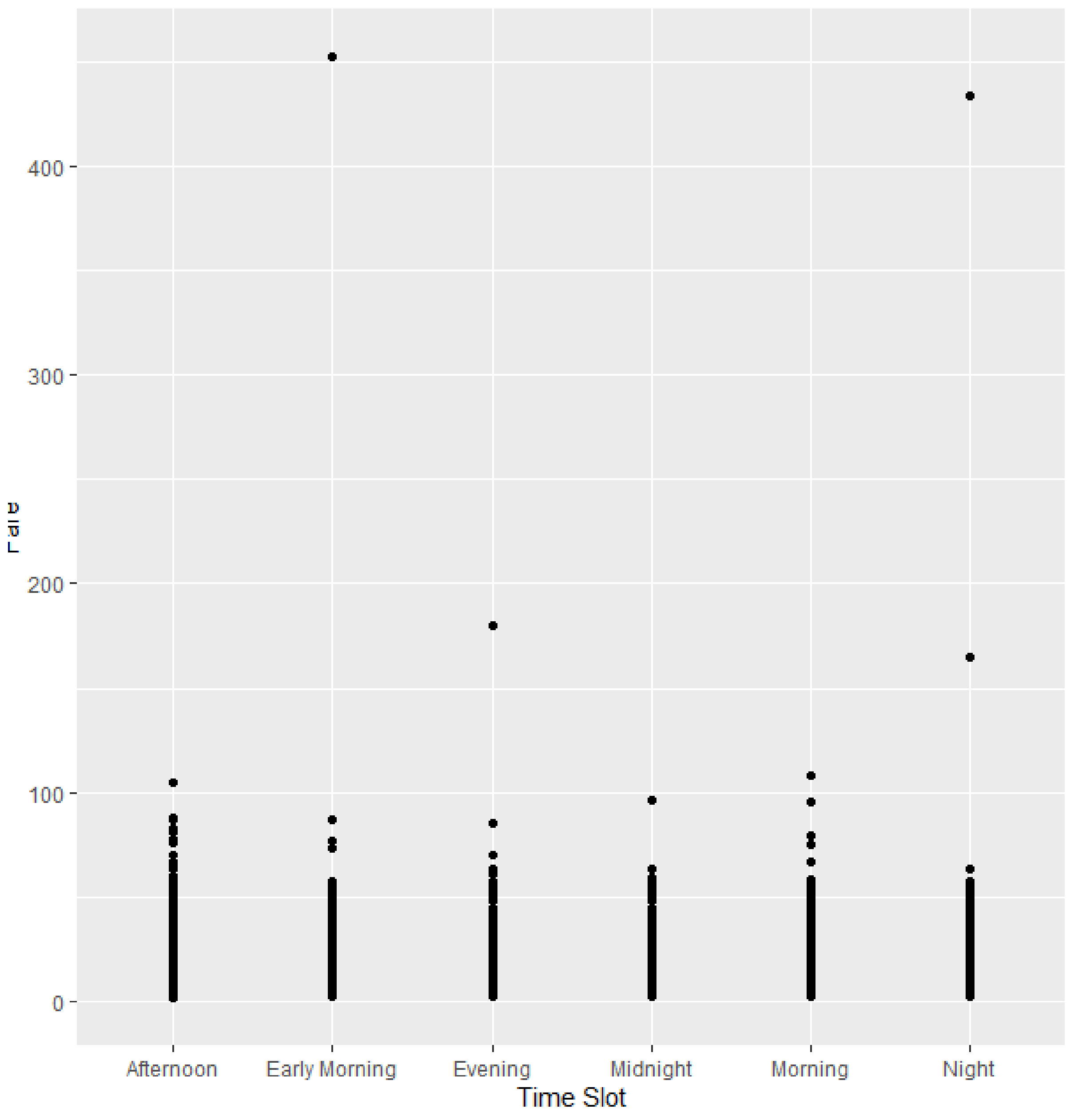
<u>For 'pickup_datetime' variable</u>:

We will use this timestamp variable to create new variables.
New features will be year, month, day_of_week, hour.
'year' will contain only years from pickup_datetime. For ex. 2009, 2010, 2011, etc.
'month' will contain only months from pickup_datetime. For ex. 1 for January, 2 for February, etc.
'day_of_week' will contain only week from pickup_datetime. For ex. 1 which is for Monday,2 for Tuesday,etc.
'hour' will contain only hours from pickup_datetime. For ex. 1, 2, 3, etc.



Time of day
count
3000
2000
1000
0
Afternoon
Early Morning
Evening
Midnight
Morning
Night
pickup_Duration

Fare as per time



As we have now these new variables we will categorize them to new variables like Session from hour column, seasons from month column, week:weekday/weekend from day_of_week variable.

So, session variable which will contain categories—morning, afternoon, evening, night_PM, night_AM.
Seasons variable will contain categories—spring, summer, fall, winter.
Week will contain categories—weekday, weekend.

For ‘Latitudes’ and ‘Longitudes’ variables:

As we have latitude and longitude data for pickup and dropoff, we will find the distance the cab travelled from pickup and dropoff location.

We will use haversine

Columns in training data after feature engineering:

```
Index(['fare_amount', 'passenger_count_2', 'passenger_count_3',
       'passenger_count_4', 'passenger_count_5', 'passenger_count_6',
       'season_spring', 'season_summer', 'season_winter', 'week_weekend',
       'session_evening', 'session_morning', 'session_night_AM',
       'session_night_PM', 'year_2010', 'year_2011', 'year_2012', 'year_2013',
       'year_2014', 'year_2015', 'geodesic'],
      dtype='object')
```

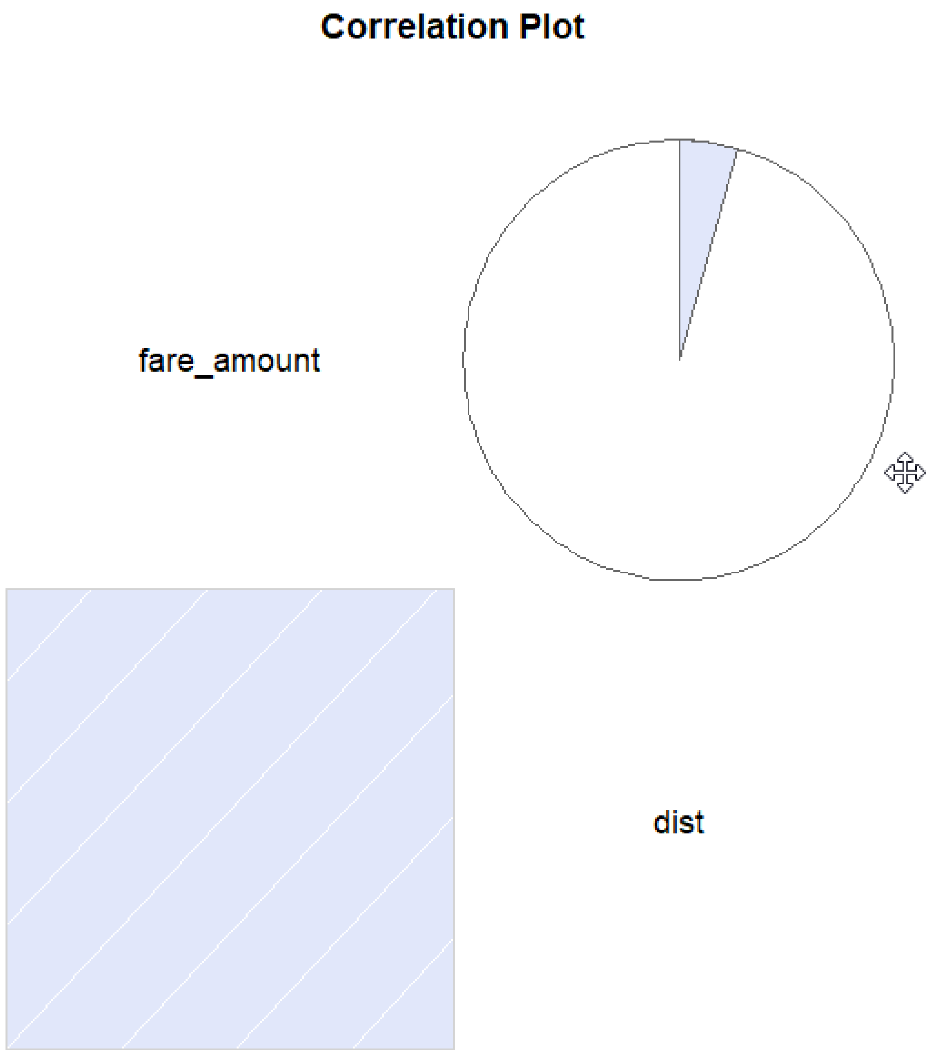
2.6 Feature Selection

In this step we would allow only to pass relevant features to further steps. We remove irrelevant features from the dataset. We do this by some statistical techniques, like we look for features which will not be helpful in predicting the target variables. In this dataset we have to predict the fare_amount.

Further below are some types of test involved for feature selection:

1. **Correlation analysis** – This requires only numerical variables. Therefore, we will filter out only numerical variables and feed it to correlation analysis. We do this by plotting correlation plot for all numerical variables. There should be no correlation between independent variables but there should be high correlation between independent variable and dependent variable. So, we plot the correlation plot. we can see that in correlation plot faded colour like skin colour indicates that 2 variables are highly correlated with each other. As the colour fades correlation values increases.
   From below correlation plot we see that:
   - 'fare_amount' and 'geodesic' are very highly correlated with each other.
   - As fare_amount is the target variable and ‘geodesic’ is independent variable we will keep ‘geodesic’ because it will help to explain variation in fare_amount.

Correlation Plot:



3 **Analysis of Variance(Anova) Test** –

I. It is carried out to compare between each group in a categorical variable.

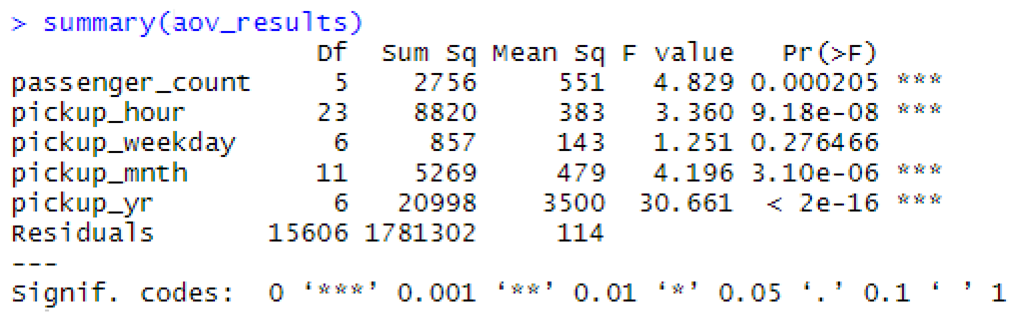
II. ANOVA only lets us know the means for different groups are same or not. It doesn't help us identify which mean is different.

Hypothesis testing:

- **Null Hypothesis**: mean of all categories in a variable are same.
- **Alternate Hypothesis**: mean of at least one category in a variable is different.

- 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.

Below is the anova analysis table for each categorical variable:



```
> summary(aov_results)
                   Df  Sum Sq Mean Sq F value   Pr(>F)    
passenger_count     5    2756     551   4.829 0.000205 ***
pickup_hour        23    8820     383   3.360 9.18e-08 ***
pickup_weekday      6     857     143   1.251 0.276466    
pickup_mnth        11    5269     479   4.196 3.10e-06 ***
pickup_yr           6   20998    3500  30.661  < 2e-16 ***
Residuals       15606 1781302     114                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

2.7 Feature Scaling

Data Scaling methods are used when we want our variables in data to scaled on common ground. It is performed only on continuous variables.

Linear Models assume that the data you are feeding are related in a linear fashion, or can be measured with a linear distance metric.

Also, our independent numerical variable ‘geodesic’ is not distributed normally so we had chosen normalization over standardization.

- We have checked variance for each column in dataset before Normalisation
- High variance will affect the accuracy of the model. So, we want to normalise that variance. Graphs based on which standardization was chosen:

Splitting Dataset

a) We have used sklearn’s train_test_split() method to divide whole Dataset into train and validation datset.
b) 25% is in validation dataset and 75% is in training data.
c) 11745 observations in training and 3915 observations in validation dataset.
d) We will test the performance of model on validation datset.
e) The model which performs best will be chosen to perform on test dataset provided along with original train dataset.
f) X_train y_train--are train subset.
g) X_test y_test--are validation subset.

4.Model Development

Our problem statement wants us to predict the fare_amount. This is a Regression problem. So, we are going to build regression models on training data and predict it on test data. In this project I have built models using 3 Regression Algorithms:

1. Linear Regression
2. Decision Tree
3. Random Forest

We will evaluate performance on validation dataset which was generated using Sampling. We Will deal with specific error metrics like –
Regression metrics for our Models:

- MAE(Mean Absolute Error)
- MAPE(Mean Absolute Percentage Error)

Here, we will evaluate the performance of different Regression models based on different Error Metrics

1. Multiple Linear Regressions: Here we received MAPE of 45% .Here accuracy is 55% which is very less. We will not consider this model.

2. Decision Tree Regression: Here we received MAPE of 22% .Here accuracy is 78% which is a good percentage.

3. Random Forest Regression: Here we received MAPE of 23% .Here accuracy is 77% which is less to Decision tree accuracy. We tried with changing the tree numbers but above Accuracy were best among them.

5. R Code

```
# Cab Fare Prediction

rm(list = ls())
setwd("G:/EdwisorAssignments/Cab Fare Prediction")

# #loading Libraries
x = c("ggplot2", "corrgram", "DMwR", "usdm", "caret", "randomForest", "e1071",
      "DataCombine", "doSNOW", "inTrees", "rpart.plot",
"rpart",'MASS','xgboost','stats','plyr','dplyr',"corrgram","DataCombine")


install.packages("DMwR")
install.packages(c('devtools','curl'))
library('devtools')
#load Packages
```

```
lapply(x, require, character.only = TRUE)
rm(x)


# loading datasets
train = read.csv("train_cab.csv", header = T, na.strings = c(" ", "", "NA"))
test = read.csv("test.csv", na.strings = c(" ", "", "NA"))
test_pickup_datetime = test["pickup_datetime"]
# Structure of data
str(train)
str(test)
summary(train)
summary(test)
head(train,5)
head(test,5)

#############                    Exploratory Data Analysis
#######################
# Changing the data types of variables
train$fare_amount = as.numeric(as.character(train$fare_amount))
train$passenger_count=round(train$passenger_count)

### Removing values which are not within desired range(outlier) depending upon basic
understanding of dataset.

# 1.Fare amount has a negative value, which doesn't make sense. A price amount cannot be -ve
and also cannot be 0.
#Also 2 observations have values 54343 and 4343 which is not possible for cab fare.So we will
remove these fields.

train$fare_amount[order(-train$fare_amount)]
train[which(train$fare_amount < 1 ),]

train = train[-which(train$fare_amount < 1 ),]
train = train[-which(train$fare_amount > 1000 ),]

summary(train)

#2.Passenger_count variable
count(train[which(train$passenger_count >6),])
count(train[which(train$passenger_count <1),])

# so 20 observations of passenger_count is consistenly above from 6,7,8,9,10
passenger_counts, let's check them.
train[which(train$passenger_count >6 ),]
```

```
train[which(train$passenger_count < 1 ),]

# We will remove these observations which are above 6 value because a cab cannot hold these
number of passengers.
train = train[-which(train$passenger_count < 1 ),]
train = train[-which(train$passenger_count > 6),]

# 3.Latitudes range from -90 to 90.Longitudes range from -180 to 180.Removing which does
not satisfy these ranges

nrow(train[which(train$pickup_longitude >180 ),])
nrow(train[which(train$pickup_longitude < -180 ),])
nrow(train[which(train$pickup_latitude > 90 ),])
nrow(train[which(train$pickup_latitude < -90 ),])
nrow(train[which(train$dropoff_longitude > 180 ),])
nrow(train[which(train$dropoff_longitude < -180 ),])
nrow(train[which(train$dropoff_latitude < -90 ),])
nrow(train[which(train$dropoff_latitude > 90 ),])

# There's only one outlier which is in variable pickup_latitude.So we will remove it with nan.
# Also we will see if there are any values equal to 0.

nrow(train[which(train$pickup_longitude == 0 ),])
nrow(train[which(train$pickup_latitude == 0 ),])
nrow(train[which(train$dropoff_longitude == 0 ),])
nrow(train[which(train$pickup_latitude == 0 ),])
# there are values which are equal to 0. we will remove them.
train = train[-which(train$pickup_latitude > 90),]
train = train[-which(train$pickup_longitude == 0),]
train = train[-which(train$dropoff_longitude == 0),]

################Pickup_Datetime#####

train$pickup_datetime=as.Date(as.character(train$pickup_datetime))

str(train)

# Make a copy
df=train
# train=df

#############                    Missing Value Analysis            #############
missing_val = data.frame(apply(train,2,function(x){sum(is.na(x))}))
missing_val$Columns = row.names(missing_val)
names(missing_val)[1] =  "Missing_percentage"
```

```
missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(train)) * 100
missing_val = missing_val[order(-missing_val$Missing_percentage),]
row.names(missing_val) = NULL
missing_val = missing_val[,c(2,1)]
missing_val

################ Bar graph of passenger count
#############################################
PassengerCount=ggplot(data = train, aes(x =passenger_count)) + ggtitle("passenger count") +
geom_bar()
gridExtra::grid.arrange(PassengerCount)

unique(train$passenger_count)
unique(test$passenger_count)

train[,'passenger_count'] = factor(train[,'passenger_count'], labels=(1:6))
test[,'passenger_count'] = factor(test[,'passenger_count'], labels=(1:6))

# 1.For Passenger_count:
# Actual value = 1
# Mode = 1
# KNN = 1
train$passenger_count[1000]
train$passenger_count[1000] = NA

getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Mode Method
getmode(train$passenger_count)
train$passenger_count[is.na(train$passenger_count)] = getmode(train$passenger_count)
sum(is.na(train$passenger_count))

train= knnImputation(train,k=4)

# 2.For fare_amount:
# Actual value = 10.5,
# Mean = 11.366,
# Median = 8.5,
# KNN = 10.833

train$fare_amount[1000]
train$fare_amount[1000]= NA
```

```
# Mean Method
mean(train$fare_amount, na.rm = T)

#Median Method
median(train$fare_amount, na.rm = T)

# kNN Imputation
train = knnImputation(train, k = 3)
train$fare_amount[1000]
train$fare_amount[1000]= NA

train$passenger_count[1000]
sapply(train, sd, na.rm = TRUE)

sum(is.na(train))
str(train)
summary(train)

df1=train
# train=df1
######################                    Outlier Analysis             ##################

# We Will do Outlier Analysis only on Fare_amount just for now and we will do outlier analysis
after feature engineering laitudes and longitudes.
# Boxplot for fare_amount

pl1 = ggplot(train,aes(x = factor(passenger_count),y = fare_amount))
pl1 + geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,outlier.size=1,
notch=FALSE)+ylim(0,100)

# Removing the outliers
vals = train[,"fare_amount"] %in% boxplot.stats(train[,"fare_amount"])$out
summary(vals)

train = train[which(!train$fare_amount %in% vals),]

#lets check the NA's
sum(is.na(train$fare_amount))

#Imputing with KNN
train = knnImputation(train,k=3)

# lets check the missing values
sum(is.na(train$fare_amount))
```

```
str(train)
df2=train

##Histogram##



# train=df2
#################             Feature Engineering
##########################
# 1.Feature Engineering for timestamp variable
# we will derive new features from pickup_datetime variable
# new features will be year,month,day_of_week,hour
#Convert pickup_datetime from factor to date time
train$pickup_date = as.Date(as.character(train$pickup_datetime))
train$pickup_weekday = as.factor(format(train$pickup_date,"%u"))# Monday = 1
train$pickup_mnth = as.factor(format(train$pickup_date,"%m"))
train$pickup_yr = as.factor(format(train$pickup_date,"%Y"))
pickup_time = strptime(train$pickup_datetime,"%Y-%m-%d %H:%M:%S")
train$pickup_hour = as.factor(format(pickup_time,"%H"))

#Add same features to test set
test$pickup_date = as.Date(as.character(test$pickup_datetime))
test$pickup_weekday = as.factor(format(test$pickup_date,"%u"))# Monday = 1
test$pickup_mnth = as.factor(format(test$pickup_date,"%m"))
test$pickup_yr = as.factor(format(test$pickup_date,"%Y"))
pickup_time_test = strptime(test$pickup_datetime,"%Y-%m-%d %H:%M:%S")
test$pickup_hour = as.factor(format(pickup_time_test,"%H"))

sum(is.na(train))# there were 5 'na' in pickup_datetime which created na's in above feature
engineered variables.
train = na.omit(train) # we will remove that 1 row of na's

train = subset(train,select = -c(pickup_datetime,pickup_date))
test = subset(test,select = -c(pickup_datetime,pickup_time_test))

# 2.Calculate the distance travelled using longitude and latitude
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
}
# Using haversine formula to calculate distance fr both train and test
train$dist =
haversine(train$pickup_longitude,train$pickup_latitude,train$dropoff_longitude,train$dropoff
_latitude)
test$dist =
haversine(test$pickup_longitude,test$pickup_latitude,test$dropoff_longitude,test$dropoff_lat
itude)

# We will remove the variables which were used to feature engineer new variables
train = subset(train,select = -
c(pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_latitude))
test = subset(test,select = -
c(pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_latitude))

str(train)
summary(train)

df3=train
#train=df3
# Data Visualization #
train$pickup_Duration = as.character(train$pickup_hour)

train$pickup_Duration[train$pickup_Duration %in% c("00","01","02","03")] = "Midnight"
train$pickup_Duration[train$pickup_Duration %in% c("04","05","06","07")] = "Early Morning"
train$pickup_Duration[train$pickup_Duration %in% c("08","09","10","11","12")] = "Morning"
train$pickup_Duration[train$pickup_Duration %in% c("13","14","15","16")] = "Afternoon"
train$pickup_Duration[train$pickup_Duration %in% c("17","18","19","20")] = "Evening"
train$pickup_Duration[train$pickup_Duration %in% c("21","22","23","24")] = "Night"

train$pickup_Duration=as.factor(train$pickup_Duration)

TimeBar = ggplot(data = train, aes(x = pickup_Duration))+ geom_bar()+ ggtitle("Time of day" )
gridExtra::grid.arrange(TimeBar)

scat1 = ggplot(data = train, aes(y =fare_amount, x = pickup_Duration)) + ggtitle("Fare as per
time") + geom_point() + ylab("Fare") + xlab("Time Slot")
```

```
gridExtra::grid.arrange(scat1)

scat2 = ggplot(data = train, aes(y =fare_amount, x = dist)) + ggtitle("Fare as per time") +
geom_point() + ylab("Fare") + xlab("Time Slot")
gridExtra::grid.arrange(scat2)

################                          Feature selection              ###################
numeric_index = sapply(train,is.numeric) #selecting only numeric

numeric_data = train[,numeric_index]

cnames = colnames(numeric_data)
#Correlation analysis for numeric variables
corrgram(train[,numeric_index],upper.panel=panel.pie, main = "Correlation Plot")

#ANOVA for categorical variables with target numeric variable

#aov_results = aov(fare_amount ~ passenger_count * pickup_hour * pickup_weekday,data =
train)
aov_results = aov(data = train,fare_amount ~ passenger_count + pickup_hour +
pickup_weekday + pickup_mnth + pickup_yr)

summary(aov_results)

# pickup_weekdat has p value greater than 0.05
train = subset(train,select=-pickup_weekday)

#remove from test set
test = subset(test,select=-pickup_weekday)

##################################          Feature Scaling
################################################
#Normality check
# qqnorm(train$fare_amount)
# histogram(train$fare_amount)

#Normalisation
df4=train
#train=df4

print('dist')
train[,'dist_norm'] = (train[,'dist'] - min(train[,'dist']))/
  (max(train[,'dist'] - min(train[,'dist'])))

train[,'Fare_norm'] = (train[,'fare_amount'] - min(train[,'fare_amount']))/
```

```
  (max(train[,'fare_amount'] - min(train[,'fare_amount'])))


summary(train)

scat3 = ggplot(data = train, aes(y =Fare_norm, x = dist_norm)) + ggtitle("Fare as per time") +
geom_point() + ylab("Fare") + xlab("Time Slot")
gridExtra::grid.arrange(scat3)

scat3 = ggplot(data = train, aes(y =fare_amount, x = dist_norm)) + ggtitle("Fare as per time") +
geom_point() + ylab("Fare") + xlab("Time Slot")
gridExtra::grid.arrange(scat3)

# #check multicollearity
# library(usdm)
# vif(train[,-1])
#
# vifcor(train[,-1], th = 0.9)

##Scatter plot#
scat2 = ggplot(data = train, aes(y =fare_amount, x = dist)) + ggtitle("distance and fare") +
geom_point() + ylab("Fare") + xlab("Time Slot")
gridExtra::grid.arrange(scat2)

################### Splitting train into train and validation subsets ###################
set.seed(1000)
tr.idx = createDataPartition(train$fare_amount,p=0.75,list = FALSE) # 75% in trainin and 25% in
Validation Datasets
train_data = train[tr.idx,]
test_data = train[-tr.idx,]

train=train[-6]

rmExcept(c("test","train","df",'df1','df2','df3','test_data','train_data','test_pickup_datetime'))
rmExcept(c('df3'))
###################Model Selection################
#Error metric used to select model is RMSE

#############          Linear regression            #################
lm_model = lm(fare_amount ~.,data=train_data)

summary(lm_model)
str(train_data)
plot(lm_model$fitted.values,rstandard(lm_model),main = "Residual plot",
     xlab = "Predicted values of fare_amount",
    ylab = "standardized residuals")


lm_predictions = predict(lm_model,test_data[,2:6])

qplot(x = test_data[,1], y = lm_predictions, data = test_data, color = I("blue"), geom = "point")

regr.eval(test_data[,1],lm_predictions)
# mae         mape
# 3.5303114   0.4510407



#############                     Decision Tree          #####################

Dt_model = rpart(fare_amount ~ ., data = train_data, method = "anova")

summary(Dt_model)
#Predict for new test cases
predictions_DT = predict(Dt_model, test_data[,2:6])

qplot(x = test_data[,1], y = predictions_DT, data = test_data, color = I("blue"), geom = "point")

regr.eval(test_data[,1],predictions_DT)
# mae       mape
# 1.8981592  0.2241461



#############                     Random forest          #####################
rf_model = randomForest(fare_amount ~.,data=train_data)

summary(rf_model)

rf_predictions = predict(rf_model,test_data[,2:6])

qplot(x = test_data[,1], y = rf_predictions, data = test_data, color = I("blue"), geom = "point")

regr.eval(test_data[,1],rf_predictions)
# mae         mape
# 1.9053850  0.2335395
```