BIKE RENTING

DECEMBER 27, 2019

RIDDHESH SAJWAN

Contents

Introduction

1. Problem Statement

The objective of this project is to form a model for the prediction of bike rental count on a daily basis of a bike renting company. This prediction model will be based on environmental factors like weather situation, temperature, humidity and windspeed. This model will allow the company to develop a pricing model according to the predicted demand of cycles based on environmental conditions.

2. Data

We will be using the following data in the table 1.2.1 and 1.2.2 to make a prediction model, which will depend on various factors like the month of the year, day of the week, environmental factors like temperature, humidity, windspeed.

Table 1.2.1 Bike Rental Count Sample Data (Columns: 3-9)

season	yr	mnth	holiday	weekday	workingday	weathersit
1	0	1	0	6	0	2
1	0	1	0	0	0	2
1	0	1	0	1	1	1
1	0	1	0	2	1	1

Table 1.2.2 Bike Rental Count Sample Data (Columns: 10-16)

temp	atemp	hum	windspeed	casual	registered	cnt
0.344167	0.363625	0.805833	0.160446	331	654	985
0.363478	0.353739	0.696087	0.248539	131	670	801
0.196364	0.189405	0.437273	0.248309	120	1229	1349
0.2	0.212122	0.590435	0.160296	108	1454	1562

We can observe from the above data that we will be using the following variables to predict count of bike rentals:

Table 1.2.3 Predictor Variables

SR. NO.	Predictor
1	season
2	yr
3	mnth
4	holiday
5	weekday
6	workingday
7	weathersit
8	temp
9	atemp
10	hum
11	windspeed

Methodology

1. Data Visualisation

This is a very important aspect for any data science project as this step enables the engineer to observe the data carried by the predictors in a much more informative way. In data visualisation we use various graphical representations to showcase the observations we have with us, which helps us to understand the future steps a bit more. We have the following kind of graphs in our report: (R code in Appendix)

a. Probability Density Function with Histogram and normal fit. (fig 2.1.1.1)
b. Histogram and Mean of Predictors (fig 2.1.1.2)
c. Boxplots for each Predictor (Before *fig 2.1.1.3* and After *fig 2.1.1.5* removal of outliers)

1.1 Probability Density Function

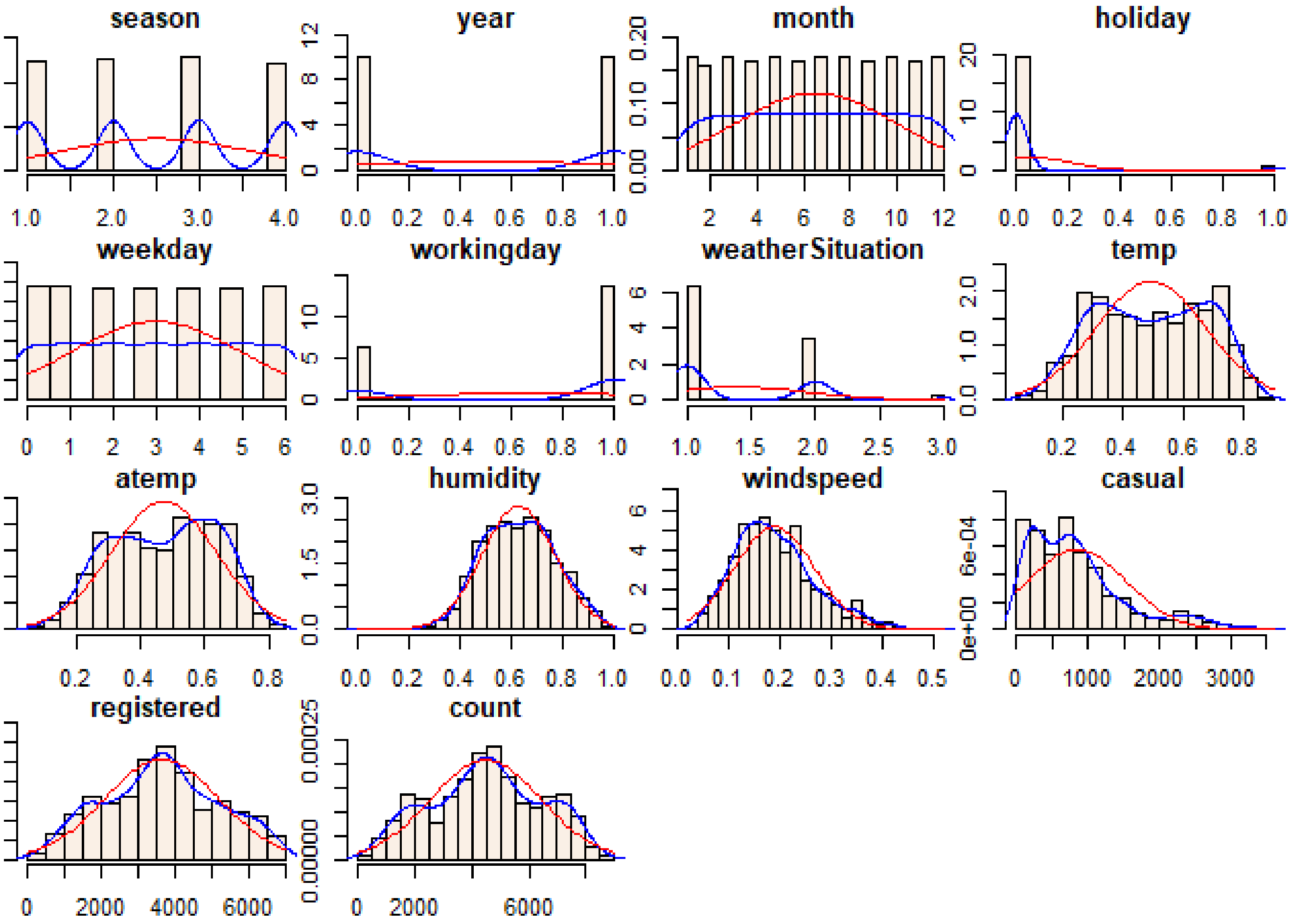
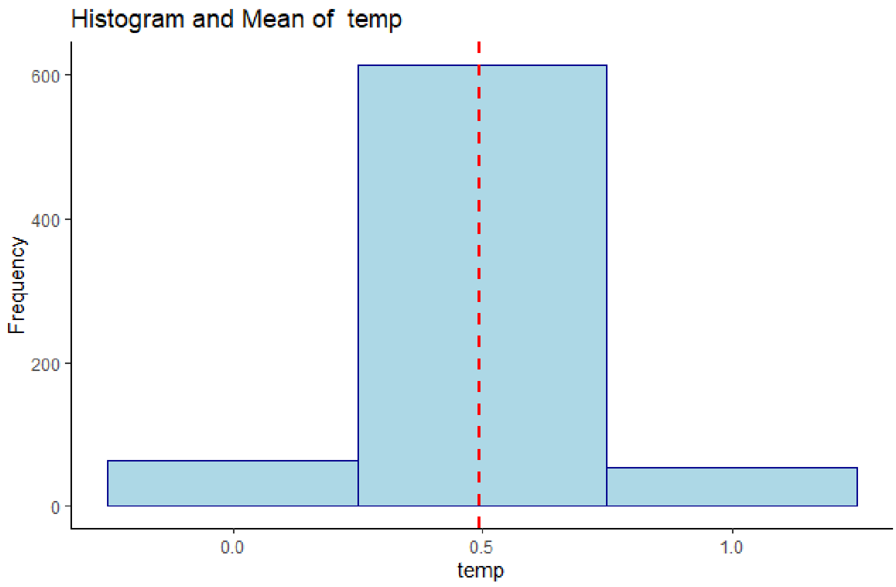
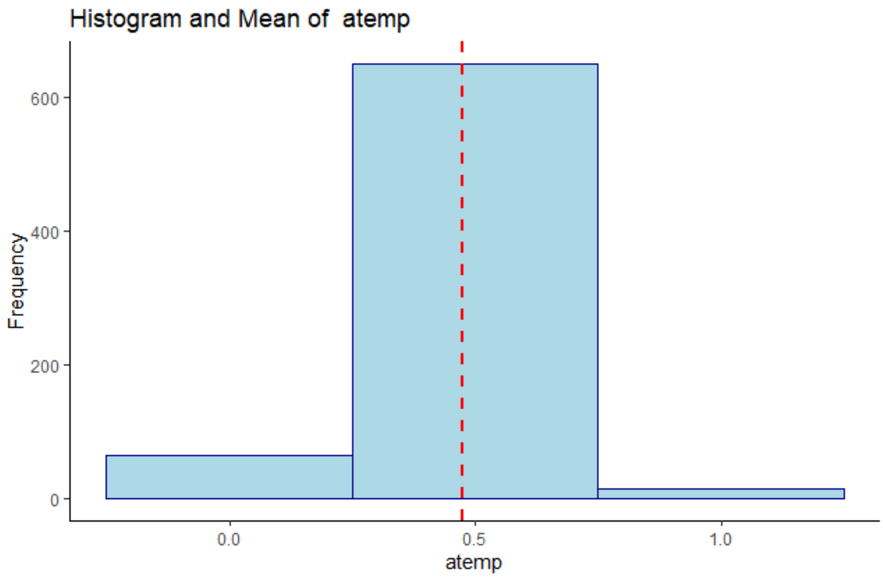
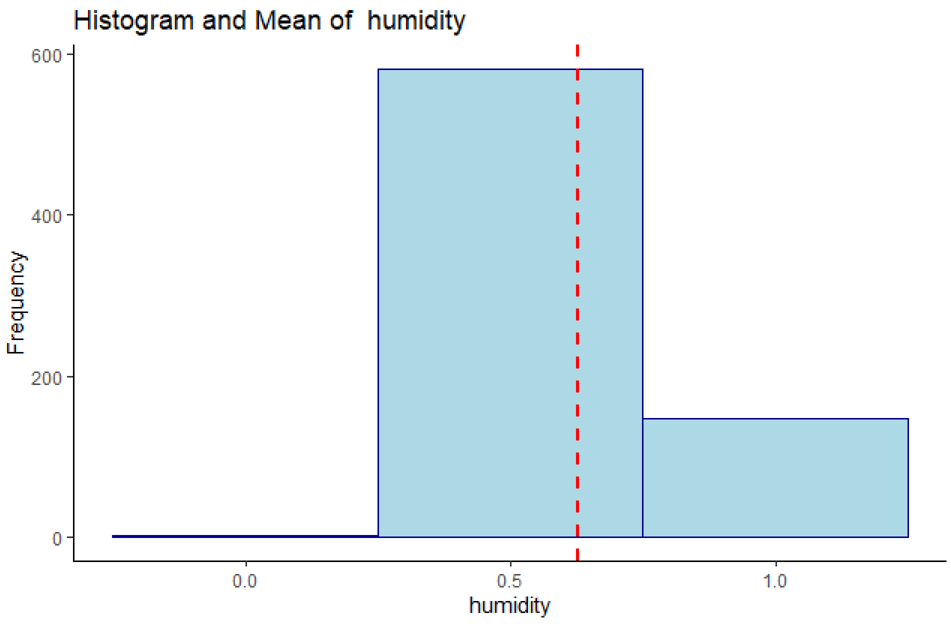


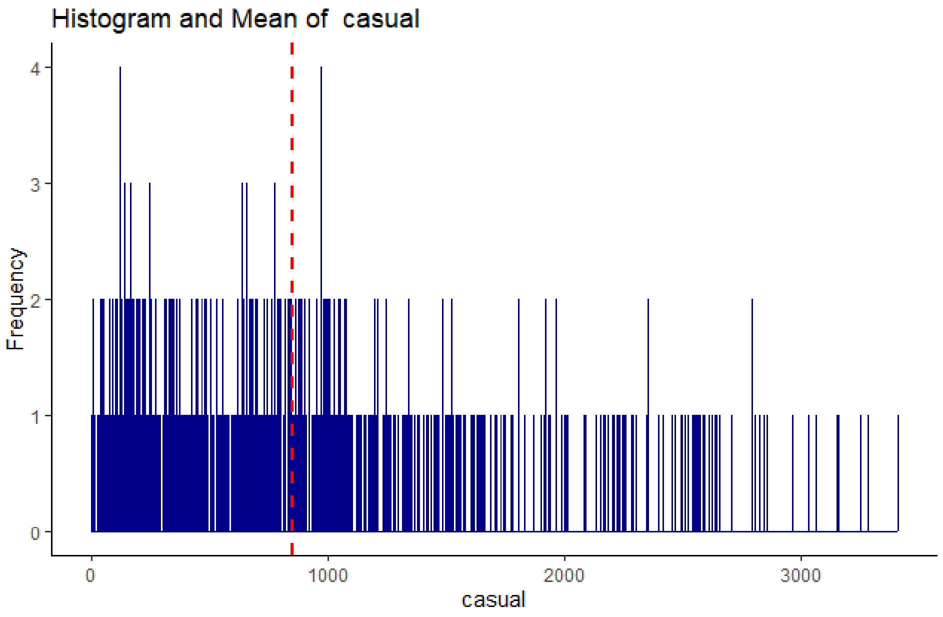
Figure 2.1.1.1 PDF with Histogram and Normal Fit

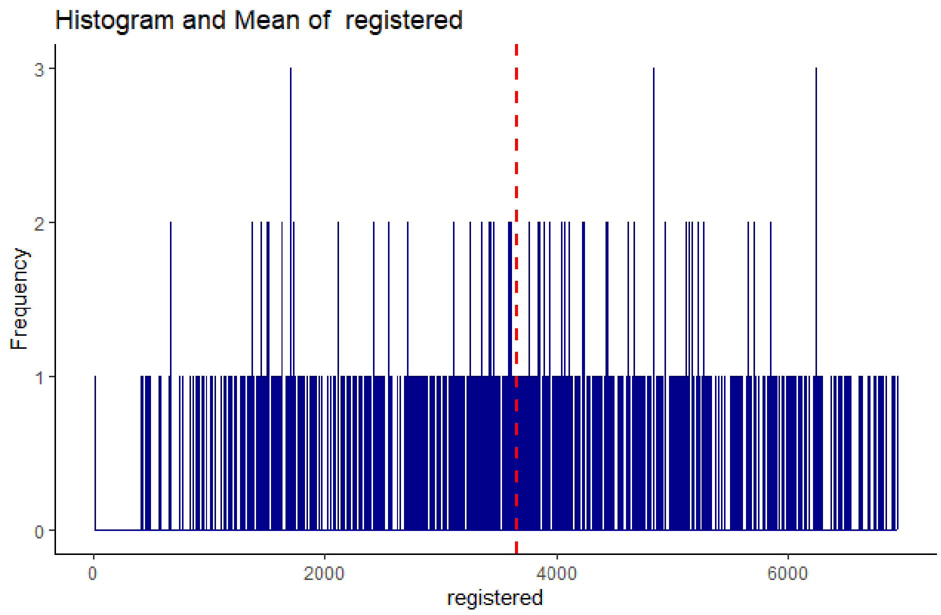
1.2 Histogram and Mean of Predictors


Histogram and Mean of temp
Frequency
600
400
200
0
0.0
0.5
1.0
temp


Histogram and Mean of atemp
Frequency
600
400
200
0
0.0
0.5
1.0
atemp


Histogram and Mean of humidity
Frequency
600
400
200
0
0.0
0.5
1.0
humidity


Histogram and Mean of casual
Frequency
4
3
2
1
0
0
1000
2000
3000
casual



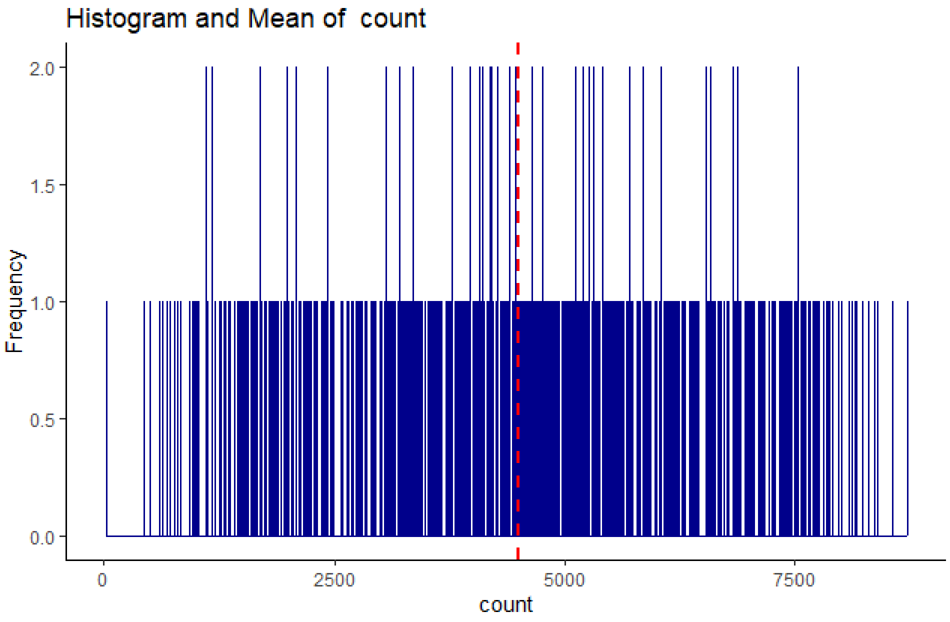
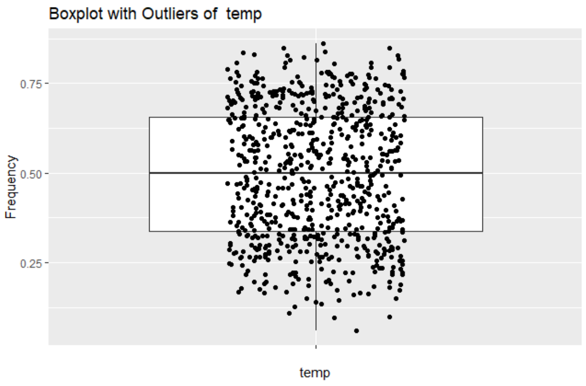
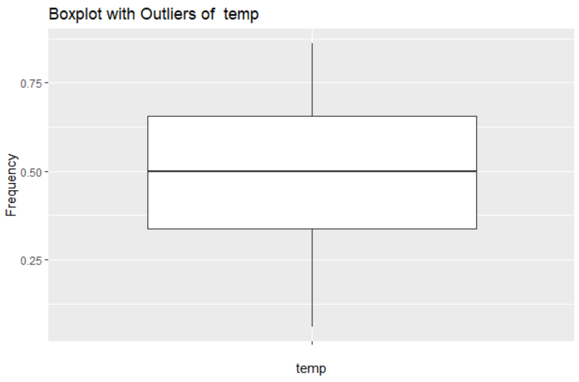
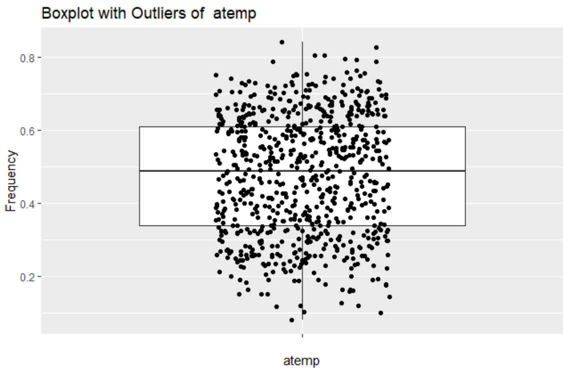


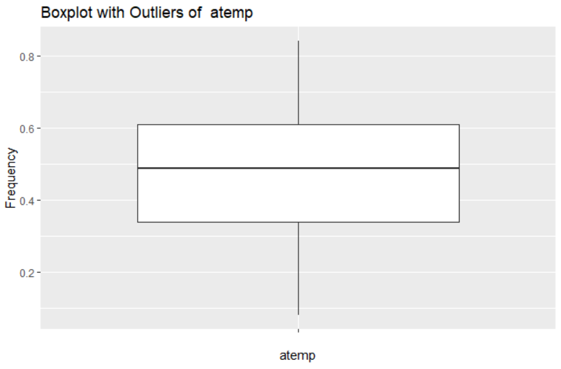
Figure 2.1.1.2 Histogram with Mean Line

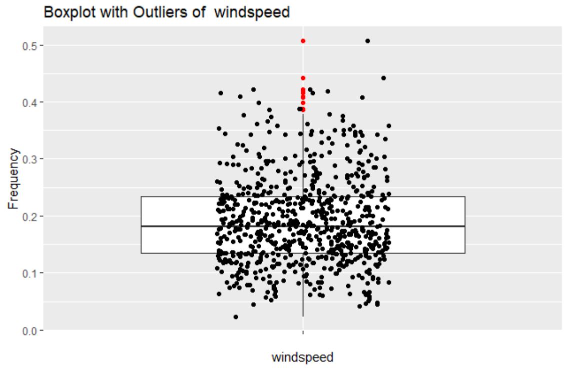
1.3 Boxplots with Outliers

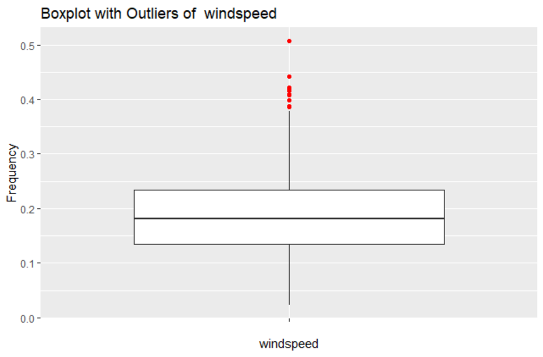


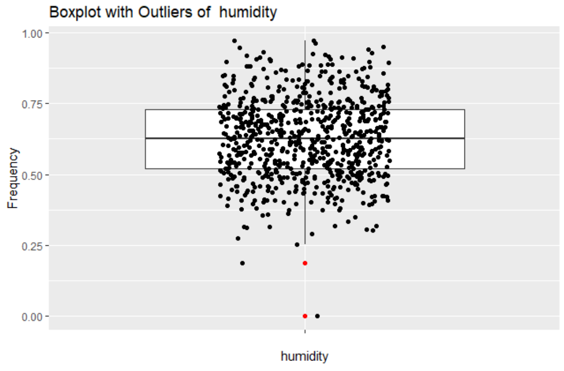


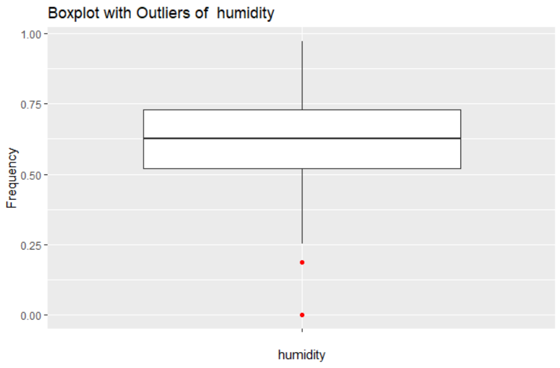












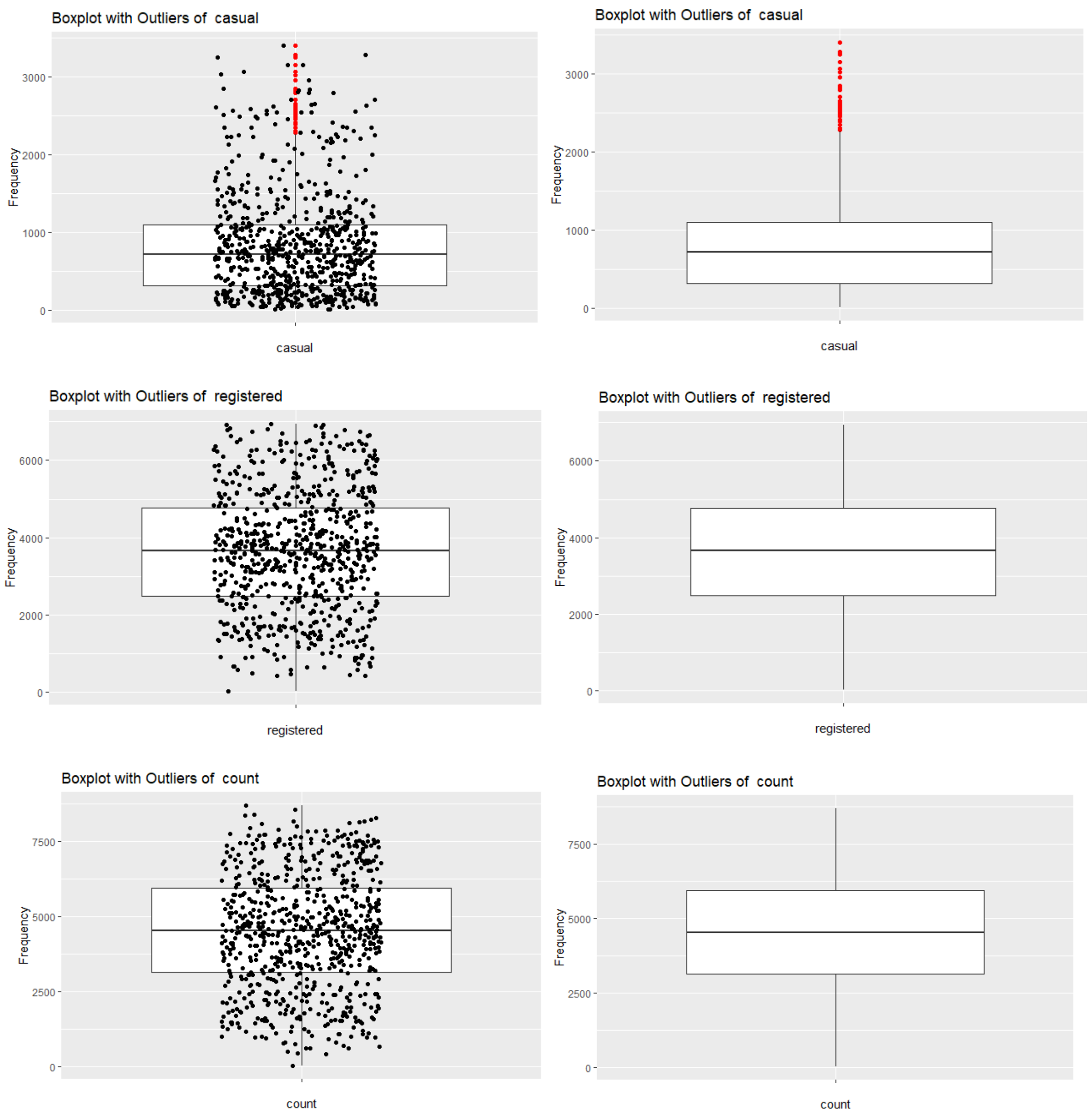
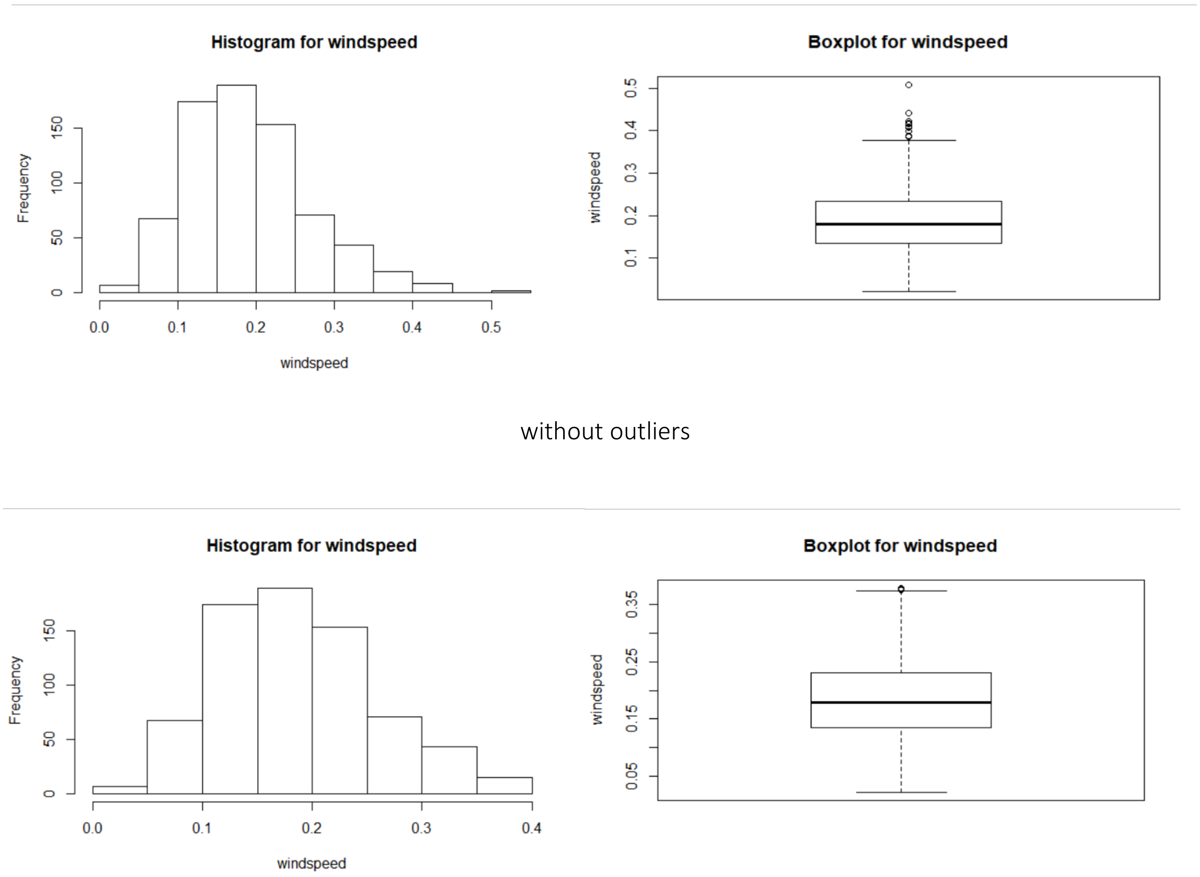


Fig 2.1.1.3 Boxplots with Outliers

The red dots observed in the above diagrams are outliers and their presence can actually affect the graphs. In the above graphs, it is quite clear that the predictor *windspeed* has a lot of outliers and removing them can totally help the outcome.

Hence, below fig 2.1.1.5 is a representation of boxplots after the removal of outliers from *windspeed* and refer fig 2.1.1.4 to see the effect of outliers on data, depicted by the use of boxplot and histogram.

1.4 Effect of Outliers



with outliers

without outliers

Fig 2.1.1.4 Effect of Outliers

1.5 Boxplots After Removal of Outliers

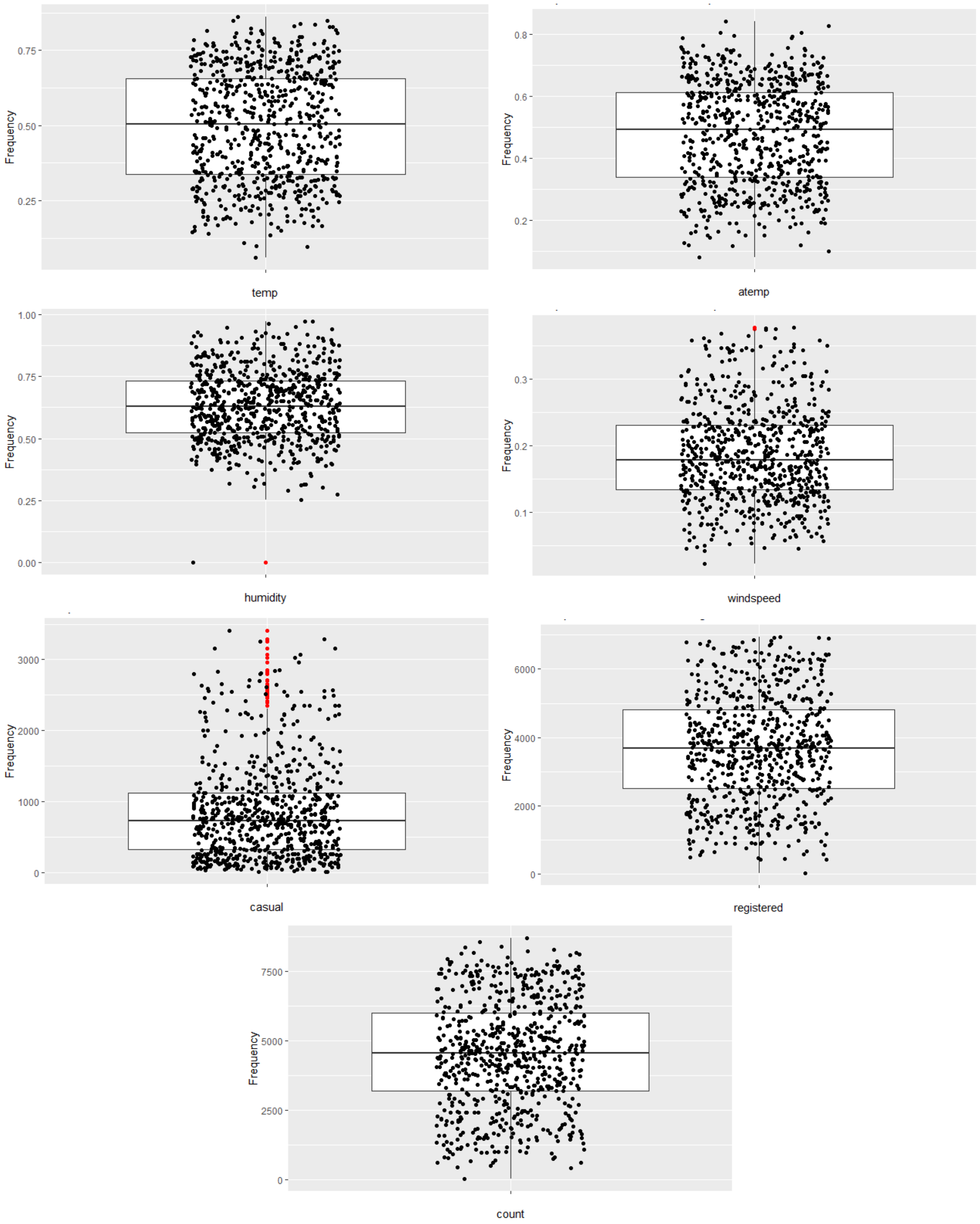


Fig 2.1.1.5 Boxplots without Outliers

2. Pre Processing

Data pre-processing is an important initial step in any data mining, data science or machine learning project. In simple terms, it is the process of converting raw data into a much readable, understandable format. The raw data usually received is always incomplete or has errors, hence data pre-processing is done to make sure that complete and correct data is sent to the deployment model to get a flawless and more accurate prediction result. The following steps take place in the process of pre-processing:

a. Missing Data Analysis
b. Outlier Analysis
c. Encoding Categorical Data
d. Standardisation of Data
e. Dimensionality Reduction
f. Splitting of Data Set

Having observed the nature of our data set from the visualisation, it is quite straight forward now to comment on the above steps.

Missing data was not observed at all during the above exercise; hence an imputation is not required. It is safe to say that the data set only contains numerical data, so no standardisation or normalisation is required.

Dimensionality reduction and splitting of data set will be done in the coming part of the project.

2.1 Outlier Analysis

"Observation which deviates so much from other observations as to arouse suspicion it was generated by a different mechanism" — Hawkins(1980)

We have clearly observed from our visualisations that some predictors like *windspeed* had a few data points that weren't as usual and hence they skewed the entire data outcome. We also proved the effects of outliers on the data using visualisations. The histograms and boxplots in the fig 1.2.4 precisely depict how the outliers affected the outcome.

2.2 Feature Selection

This is a technique used to make our dataset contain less but more contributing predictors. Given, we have a long list of predictors, it is quite possible that some of them are dependent on each other and hence do not contribute significantly in the final outcome. The process of discovering and eliminating such predictors is called feature selection. The features selected after this process become the part of our final dataset which is passed through our prediction model.

There are many ways to perform feature selection. We will be using the correlation analysis on continuous variables and chi-square test on the categorical variables to determine their inter-dependency.

Table 2.2.2.1 Continuous Variables

SR. NO.	Predictor
1	temp
2	atemp
3	hum
4	windspeed

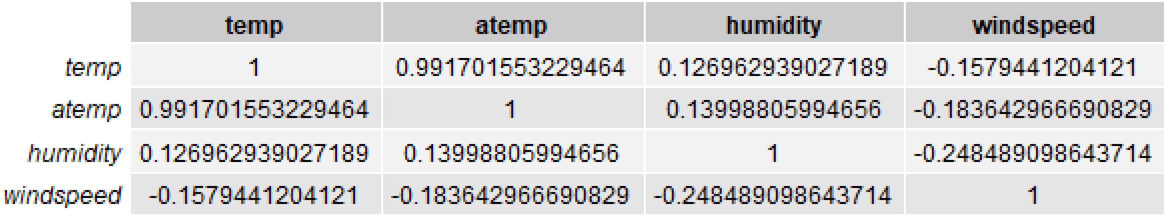
Table 2.2.2.2 Categorical Variables

SR. NO.	Predictor
1	season
2	yr
3	mnth
4	holiday
5	weekday
6	workingday
7	weathersit

Correlation Analysis

After performing correlation analysis on the continuous variables (table 2.2.2.1), we get the following correlation matrix (fig 2.2.2.3). (R code in Appendix)

Figure 2.2.2.3 Correlation Matrix



	temp	atemp	humidity	windspeed
temp	1	0.991701553229464	0.126962939027189	-0.1579441204121
atemp	0.991701553229464	1	0.13998805994656	-0.183642966690829
humidity	0.126962939027189	0.13998805994656	1	-0.248489098643714
windspeed	-0.1579441204121	-0.183642966690829	-0.248489098643714	1

On observing the above table or running a code in R to find highly correlated variables with a cut-off score of .65 (R code in Appendix), we get the variable *atemp* as dependent on *temp*. Hence, we will proceed to the prediction model with only 3 continuous variables *temp*, *humidity*, *windspeed*.

Chi-Square Test

Chi-Square test was performed on categorical variables (table 2.2.2.2) with respect to both the target variables *casual* and *registered*. Please find below (table 2.2.2.4) the result of chi-square test in the form of p-value.

Table 2.2.2.4 Chi-Square Test

	casual	registered
season	0.04521537	0.4119258
year	0.3473684	0.2666508
month	0.05691937	0.3379528
holiday	0.3962866	0.9117408
weekday	0.2487236	0.5063271
workingday	0.1716827	0.4340995
weatherSituation	0.5037164	0.4776112

The above table depicts the p-value of each categorical variable with corresponding to target variables. The chi-square test assumes the null hypothesis, under which two variables are considered independent. So, from our above table, if the p-value is less than 0.05 then the null hypothesis is accepted otherwise rejected.

In simple terms, if the p-value between two variables is less than 0.05, then both the variables are considered to be independent.
From the above data, *season* and *casual* can be considered to be independent, hence conclusively, *season* doesn't contribute much in the prediction of the target variable *casual*. So, during the prediction analysis of *casual*, *season* will be left out.

3. Modelling

In the above analysis done by us using various methods, we have been able to identify that the variable *count* is the sum of *casual* and *registered*. Hence, we only need to predict those two variables using different methods. Given, our target variables are continuous, we will be relying only on regression-based models to make our predictions. Below are the models we have used in our project.

3.1 Multiple Linear Regression

Considering that we have multiple predictors, it is safe and efficient to use multiple regression on our variables in order to get a prediction model out of them.

Output for casual (MLR)

```
Call:
lm(formula = casual ~ ., data = dsContCas)

Residuals:
Min       1Q   Median       3Q      Max
-1269.43  -212.90   -11.52   172.22  1632.66

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)       754.512    117.403   6.427 2.77e-10 ***
year              290.240     33.028   8.788  < 2e-16 ***
month               3.286      5.055   0.650 0.515979
holiday          -294.051     95.497  -3.079 0.002176 **
weekday            31.061      8.245   3.767 0.000182 ***
workingday       -859.215     36.533 -23.519  < 2e-16 ***
weatherSituation -126.126     39.034  -3.231 0.001304 **
temp             1997.595     96.607  20.678  < 2e-16 ***
humidity         -333.297    157.922  -2.111 0.035252 *
windspeed        -845.625    242.812  -3.483 0.000535 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 392 on 566 degrees of freedom
Multiple R-squared:  0.6873,   Adjusted R-squared:  0.6824
F-statistic: 138.2 on 9 and 566 DF,  p-value: < 2.2e-16
```

Output for registered (MLR)

```
Call:
lm(formula = registered ~ ., data = dsContReg)

Residuals:
    Min      1Q  Median      3Q     Max 
-4021.8  -443.1    62.6   537.8  2962.8 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)       1879.16     269.83   6.964 9.21e-12 ***
season             461.76      68.19   6.771 3.21e-11 ***
year              2021.48      75.04  26.939  < 2e-16 ***
month              -21.92      21.65  -1.013 0.311559    
holiday           -650.51     217.58  -2.990 0.002914 ** 
weekday             58.45      18.73   3.121 0.001896 ** 
workingday          67.46      83.05   0.812 0.416950    
weatherSituation  -594.60      88.68  -6.705 4.90e-11 ***
temp              5219.43     225.65  23.130  < 2e-16 ***
humidity         -1265.66     358.96  -3.526 0.000456 ***
windspeed        -2818.17     552.50  -5.101 4.63e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 890.6 on 565 degrees of freedom
Multiple R-squared:  0.7928,	Adjusted R-squared:  0.7892 
F-statistic: 216.2 on 10 and 565 DF,  p-value: < 2.2e-16
```

As we can clearly observe from the above outputs of the models in R, that the adjusted R-squared and p-value seem to be in our favour. Adjusted R-square values depicts that we can explain about 68% (for casual)/79%(for registered) of the data using multiple linear regression and the p-value is a proof that the target variable is dependent on at least one of the predictors.

3.2 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

In the coming sub-sections, you will see the output in R after running training data through a random forest model.

Output for casual (RF)

```

Call:
 randomForest(formula = casual ~ ., data = dsCatCas)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 81510.34
                    % Var explained: 83.13
```

Output for registered (RF)

```

Call:
 randomForest(formula = registered ~ ., data = dsCatReg)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 507966.4
                    % Var explained: 86.47
```

The above outputs depict a few things about the model. The number of trees used by the model to train is 500 (which is default in R) and the fact that this model has 83.13% (for casual) and 86.47% (for registered) variance explained is a good sign for us.

Conclusion

1. Model Evaluation

Now that we have already passed our training data through the prediction models, we will now determine which model should be actually used to predict our target variables. Selection of model can be done using various parameters, but in this project, we have decided to stick to *accuracy* of predictions as a measure to choose.

1.1 Root Mean Squared Error (RMSE)

To determine the accuracy of predictions, we will be using the RMSE (Root Mean Squared Error) method. Below is the formula used to calculate accuracy percentage, code of which was written in R:

$$100 - \frac{RMSE * 100}{Mean(Test\ Values\ of\ Target\ Variable)}$$

Accuracy of Multiple Linear Regression Model

```
[1] "Accuracy for casual  55.3848517251911"
```

```
[1] "Accuracy for registered  81.7503400857908"
```

Accuracy of Random Forest Model

```
[1] "Accuracy for casual  70.6430104472205"
```

```
[1] "Accuracy for registered  86.4664565388747"
```

2. Model Selection

From the above depictions of accuracy percentage, it is quite evident that the random forest model is much more accurate and hence it should be chosen to make predictions in the future.

Appendix

R Code

Data Visualisation

```
library(psych)
library(ggplot2)

#reading csv
ds <- read.csv("C:/Users/riddh/OneDrive/Documents/Edwisor/Project 1/day.csv", sep = ",")
#removing unwanted variables
dsPredictors <- ds[,3:16]
#renaming colnames
dsPredictors <- dplyr::rename(dsPredictors, 'year' = yr,
                      'month' = mnth, 'weatherSituation' = weathersit,
                      'count' = cnt, 'humidity' = hum)

#Probability Density Funciton with histogram and
multi.hist(dsPredictors, main = NULL, dcol = c("blue", "red"),
           dlty = c("solid", "solid"), bcol = "linen")



#histogram with mean line
for (i in 8:14){

  title_val <- paste("Histogram and Mean of ",colnames(dsPredictors[i]))
  hist_plot<-ggplot(dsPredictors, aes(x=dsPredictors[,i])) +
    geom_histogram(binwidth=0.5,color="darkblue", fill="lightblue",
                   linetype="solid")+
    labs(title=title_val,x=colnames(dsPredictors[i]), y = "Frequency")+
    theme_classic() +
    geom_vline(aes(xintercept=mean(dsPredictors[,i])),
               color="red", linetype="dashed", size=1)

  print(hist_plot)
}
```

Outlier Analysis

```
#outlier analysis
drawBoxP <- function(df){
  for (i in 8:14){
    title_val <- paste("Boxplot with Outliers of ",colnames(df[i])) # paste("Boxplot without Outliers of ",colnames(df[i]))
    i<-12
    box_plot <- ggplot(df, aes(x='', y= df[,i])) +
                geom_boxplot(outlier.color = "red", outlier.shape = 19,
                   outlier.size = 1.5, outlier.stroke = 0.5) +
                labs(title=title_val,x=colnames(df[i]), y = "Frequency") +
                geom_jitter(shape=16, position=position_jitter(0.2)) #with jitter
    print(box_plot)
  }
}
#boxplot with outliers
drawBoxP(dsPredictors)
#removing outliers from windspeed
unwanted_outliers <- boxplot(dsPredictors$windspeed, plot = FALSE)$out
dsOutRemoved <- dsPredictors[-which(dsPredictors$windspeed %in% unwanted_outliers ),]
drawBoxP(dsOutRemoved)

#effect of outliers

#with outliers

boxplot(dsPredictors$windspeed, main="Boxplot for windspeed",
        ylab="windspeed")
hist(dsPredictors$windspeed, main="Histogram for windspeed",
     xlab="windspeed")
#without outliers
boxplot(dsOutRemoved$windspeed, main="Boxplot for windspeed",
        ylab="windspeed")
hist(dsOutRemoved$windspeed, main="Histogram for windspeed",
     xlab="windspeed")
```

Feature Selection

```
library(mlbench)
library(caret)
library(gridExtra)

#dataframe with continuos variables
dsCont <- dsOutRemoved[,8:11]
#correlation analysis
corrMat <- cor(dsCont)

#printing out png of correlation matrix
png("correlationMatrix.png", height = 200, width = 600)
grid.table(corrMat)
dev.off()

#finding highly correlated
highCorr <- findCorrelation(corrMat, cutoff=0.65)
print(highCorr)

#dataframe with categorical variables
dsCat <- dsOutRemoved[,1:7]
#making function
chiTestFunc <- function(df,var){
  j <- grep(var,colnames(dsOutRemoved))
  for (i in 1:7){
    print(colnames(dsCat[i]))
    chiTestOut <- chisq.test(dsOutRemoved[,var], dsCat[,i])
    print(chiTestOut$p.value)}
}

#chisquare test with casual
var <- "casual"
chiTestFunc(dsCat,var)

#chisquare test with registered
var <- "registered"
chiTestFunc(dsCat,var)
remove(col)
corrMatTemp <- cor(dsPredictors)

#removing unwanted feature atemp due to corr analysis
dsOutRemoved$atemp <- NULL
```

Data Training

```
library(tidyverse)
library(randomForest)
library(caret)

#index for partition: test and train
trainIndex = createDataPartition(dsOutRemoved$count, p = .80, list = FALSE)

#multiple regression model for continuous variables

#for casual
dsContCas <- dsOutRemoved[trainIndex,1:11]
dsContCasTest <- dsOutRemoved[-trainIndex,1:11]

#removing season due to feature selection chi-test
dsContCas$season <- NULL
dsContCasTest$season <- NULL
mlrmodelCas <- lm(casual ~., data = dsContCas)
summary(mlrmodelCas)

#for registered
dsContReg <- dsOutRemoved[trainIndex,1:10]
dsContRegTest <- dsOutRemoved[-trainIndex,1:10]
dsContReg$registered <- dsOutRemoved[trainIndex,13]
dsContRegTest$registered <- dsOutRemoved[-trainIndex,13]
mlrmodelReg <- lm(registered ~., data = dsContReg)
summary(mlrmodelReg)

#random forest for categorical variables

#for casual
dsCatCas <- dsOutRemoved[trainIndex,1:11]
dsCatCasTest <- dsOutRemoved[-trainIndex,1:11]

#removing season due to feature selection chi-test
dsCatCas$season <- NULL
dsCatCasTest$season <- NULL

#keeping categorical variables as categorical
dsCatCas <- transform(dsCatCas,
                   year = as.factor(year),
                   month = as.factor(month),
                   holiday = as.factor(holiday),
                   weekday = as.factor(weekday),
                   workingday = as.factor(workingday),
                   weatherSituation = as.factor(weatherSituation))
dsCatCasTest <- transform(dsCatCasTest,
                   year = as.factor(year),
                   month = as.factor(month),
                   holiday = as.factor(holiday),
                   weekday = as.factor(weekday),
                   workingday = as.factor(workingday),
                   weatherSituation = as.factor(weatherSituation))
rfmodelCas <- randomForest(casual ~ ., data = dsCatCas)
print(rfmodelCas)

#for registered
dsCatReg <- dsOutRemoved[trainIndex,1:10]
dsCatRegTest <- dsOutRemoved[-trainIndex,1:10]
dsCatReg$registered <- dsOutRemoved[trainIndex,13]
dsCatRegTest$registered <- dsOutRemoved[-trainIndex,13]

#keeping categorical variables as categorical
dsCatReg <- transform(dsCatReg,
                   season = as.factor(season),
                   year = as.factor(year),
                   month = as.factor(month),
                   holiday = as.factor(holiday),
                   weekday = as.factor(weekday),
                   workingday = as.factor(workingday),
                   weatherSituation = as.factor(weatherSituation))
dsCatRegTest <- transform(dsCatRegTest,
                   season = as.factor(season),
                   year = as.factor(year),
                   month = as.factor(month),
                   holiday = as.factor(holiday),
                   weekday = as.factor(weekday),
                   workingday = as.factor(workingday),
                   weatherSituation = as.factor(weatherSituation))
rfmodelReg <- randomForest(registered ~ ., data = dsCatReg)
print(rfmodelReg)
```

Prediction and Accuracy

```
#predictions and accuracy for MLR

#for casual
casPredictionsMLR <- mlrmodelCas %>% predict(dsContCasTest)
casAccuracyMLR <- 100 - RMSE(casPredictionsMLR,
dsContCasTest$casual)*100/mean(dsContCasTest$casual)
print(paste("Accuracy for casual ",casAccuracyMLR))
#for registered
regPredictionsMLR <- mlrmodelReg %>% predict(dsContRegTest)
regAccuracyMLR <- 100 - RMSE(regPredictionsMLR,
dsContRegTest$registered)*100/mean(dsContRegTest$registered)
print(paste("Accuracy for registered ",regAccuracyMLR))
#predictions and accuracy for RF

#for casual
casPredictionsRF <- rfmodelCas %>% predict(dsCatCasTest)
casAccuracyRF <- 100 - RMSE(casPredictionsRF,
dsCatCasTest$casual)*100/mean(dsCatCasTest$casual)
print(paste("Accuracy for casual ",casAccuracyRF))
#for registered
regPredictionsRF <- rfmodelReg %>% predict(dsCatRegTest)
regAccuracyRF <- 100 - RMSE(regPredictionsRF,
dsCatRegTest$registered)*100/mean(dsCatRegTest$registered)
print(paste("Accuracy for registered ",regAccuracyRF))
```

References

www.geeksforgeeks.org

www.wikipedia.org

www.towardsdatascience.com

www.medium.com

www.data-flair.training

www.sthda.com

www.r-graph-gallery.com

www.machinelearningmastery.com

www.machinelearningmastery.com