# Predicting Bike Rentals

ASHISH PREMRAJ

11 JANUARY, 2019

# Contents

# Chapter 1

## Introduction

### 1.1 Problem Statement

To increase the sales in renting bikes, this project aims at building an effective model to predict bike rental counts on daily basis based on environmental and seasonal settings. Based on some of the features like seasons, work day, weekends, holiday, weather etc. we would like to predict the days we can expect high rental counts, moderate and low rental counts. This would benefit the business in increasing the sales and make effective business decisions.

### 1.2 Data

Our task is to build a regression model to predict the number of bike rentals on a daily basis based on multiple features like seasons and weather. Given below is the sample data set that we will be using to predict our bike rental counts.

```
#Load data
data_df = read.csv("day.csv")
head(data_df)

##   instant     dteday season yr mnth holiday weekday workingday weathersit
## 1       1 2011-01-01      1  0    1       0       6          0          2
## 2       2 2011-01-02      1  0    1       0       0          0          2
## 3       3 2011-01-03      1  0    1       0       1          1          1
## 4       4 2011-01-04      1  0    1       0       2          1          1
## 5       5 2011-01-05      1  0    1       0       3          1          1
## 6       6 2011-01-06      1  0    1       0       4          1          1
##       temp    atemp      hum windspeed casual registered  cnt
## 1 0.344167 0.363625 0.805833 0.1604460    331        654  985
## 2 0.363478 0.353739 0.696087 0.2485390    131        670  801
## 3 0.196364 0.189405 0.437273 0.2483090    120       1229 1349
## 4 0.200000 0.212122 0.590435 0.1602960    108       1454 1562
## 5 0.226957 0.229270 0.436957 0.1869000     82       1518 1600
## 6 0.204348 0.233209 0.518261 0.0895652     88       1518 1606
```

As you can see from the table below we have 15 variables, using which we will correctly predict our target variable.

```
##  [1] "instant"    "dteday"     "season"     "yr"         "mnth"
##  [6] "holiday"    "weekday"    "workingday" "weathersit" "temp"
## [11] "atemp"      "hum"        "windspeed"  "casual"     "registered"
```

# Chapter 2

## Methodology

### 2.1 Data Preprocessing

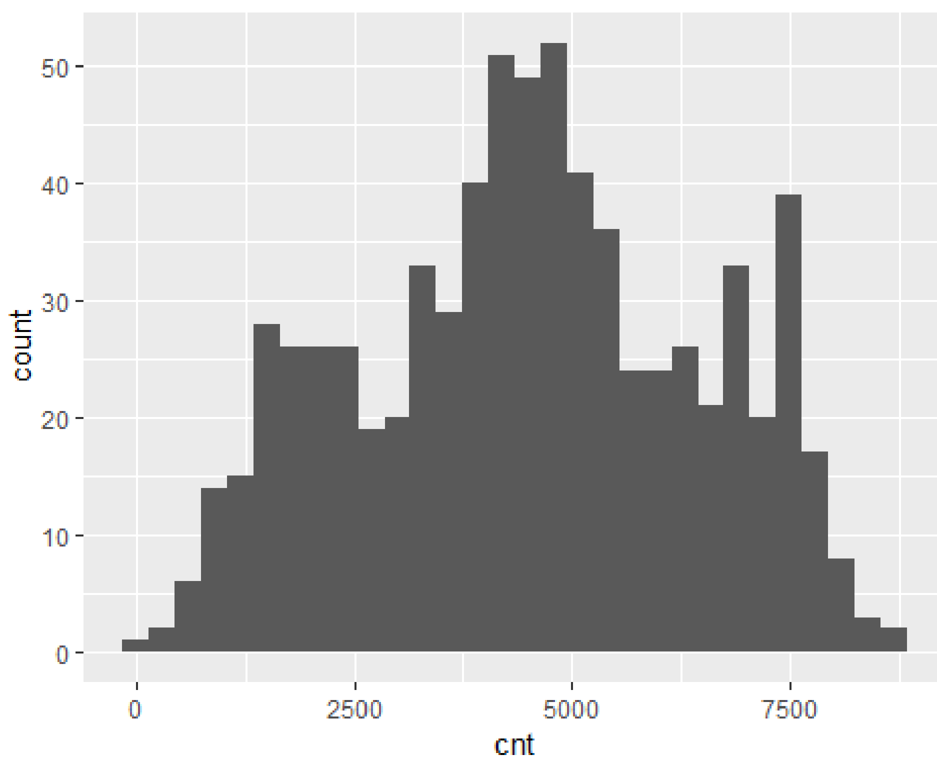
Data Preprocessing is a crucial step in data science projects to effectively evaluate our model. Data Preprocessing is a data mining technique that involves transforming raw data into a meaningful or clean data set. To do that we follow some data preprocessing steps like Exploratory Data Analysis, Outlier Analysis, Missing Value Analysis, Feature Engineering and Feature Selection.

#### 2.1.1 Exploratory Data Analysis

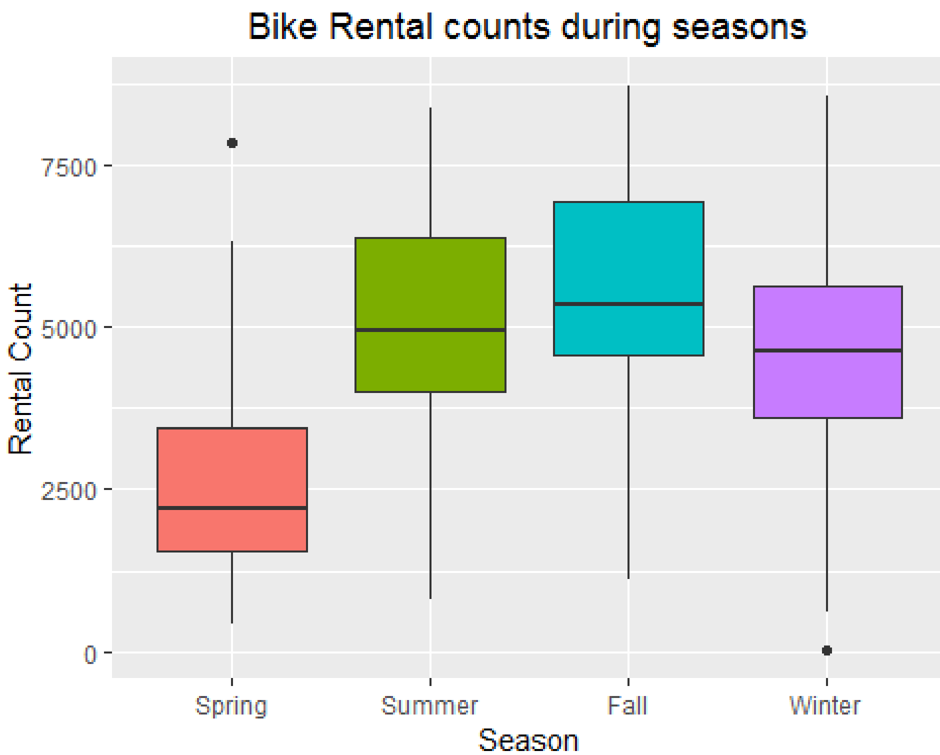
Exploratory Data Analysis is an approach of analyzing data set to summarize their main characteristics, often with visual methods. It focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, handling missing values and making transformations of variables as needed.

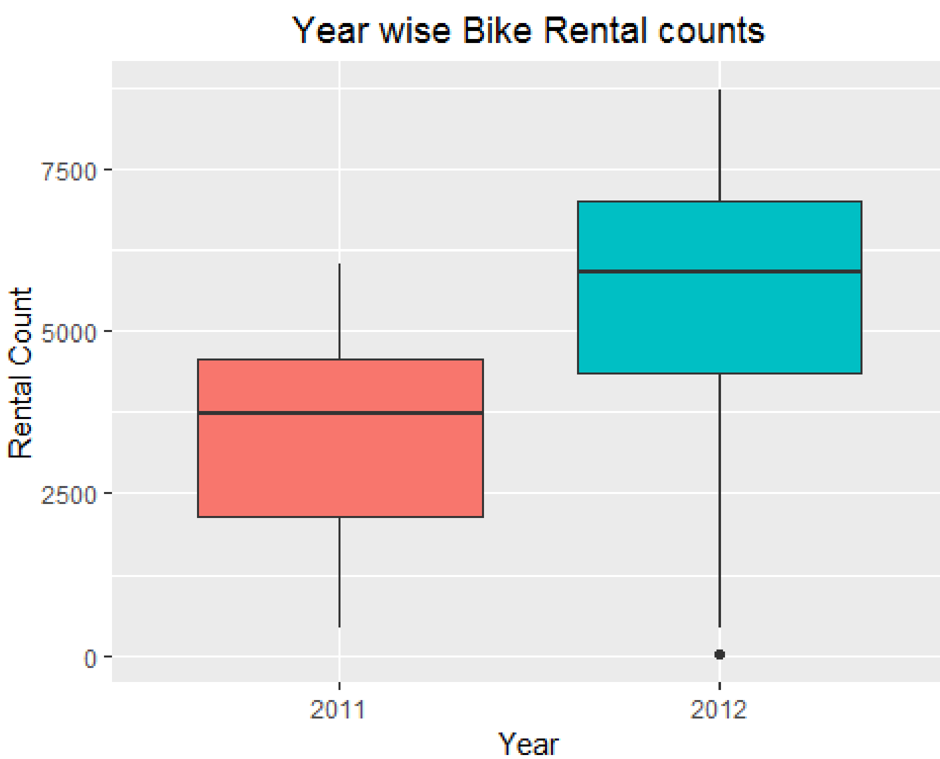
Here below we can see the distribution of our dependent variable - count

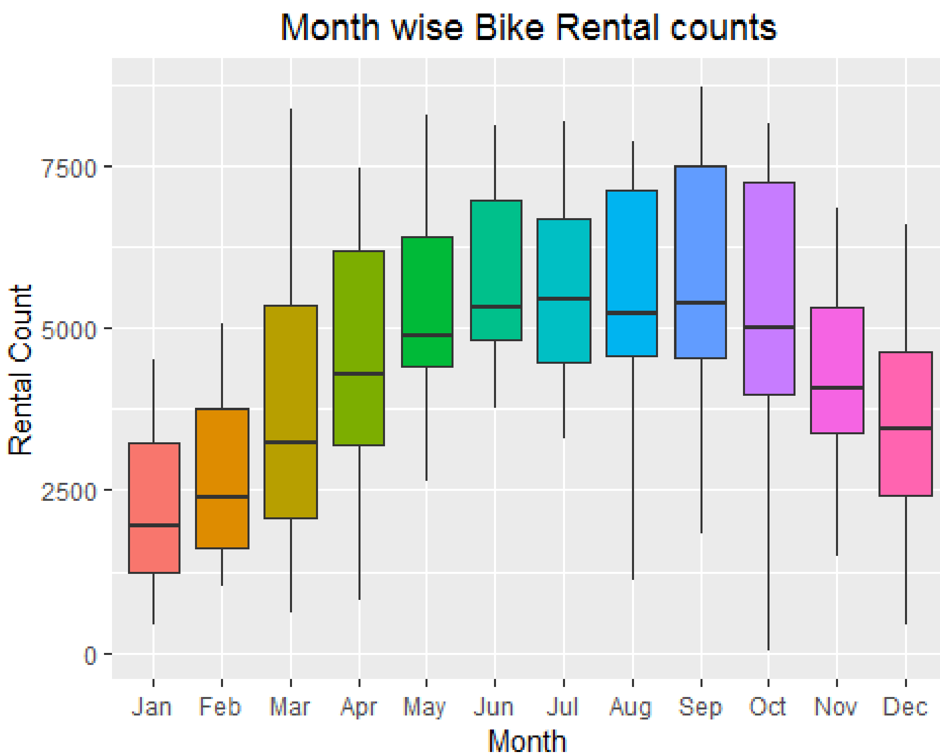
```
#Analysis on dependent variable - Count
ggplot(data = data_df, aes(cnt))+
  geom_histogram()
```



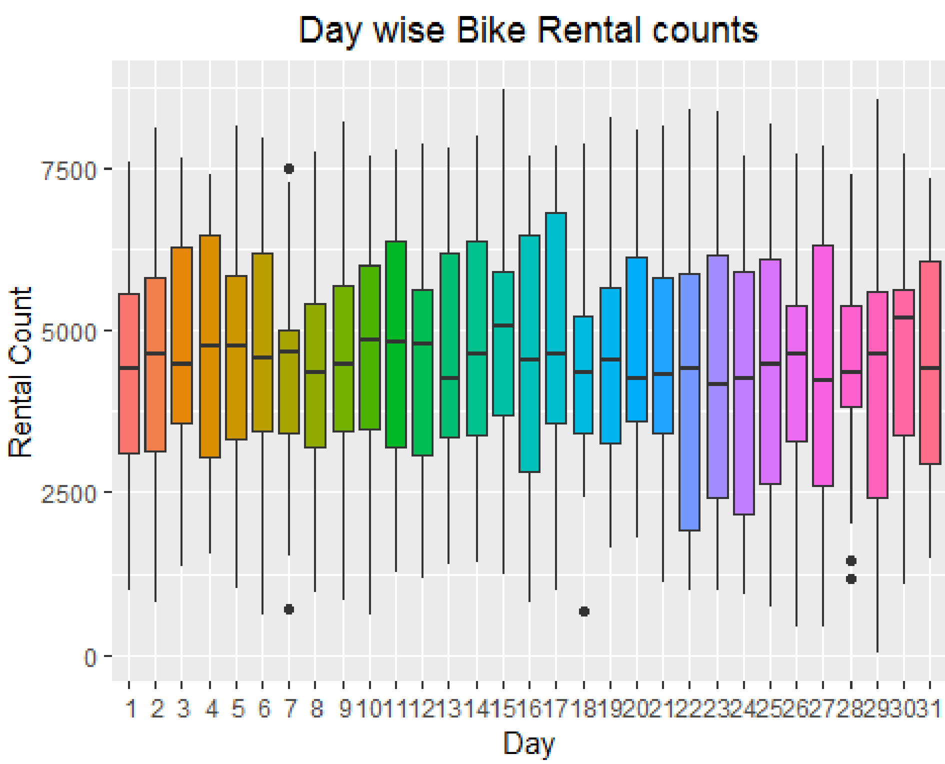
Analysis of all categorical independent variables vs dependent variable



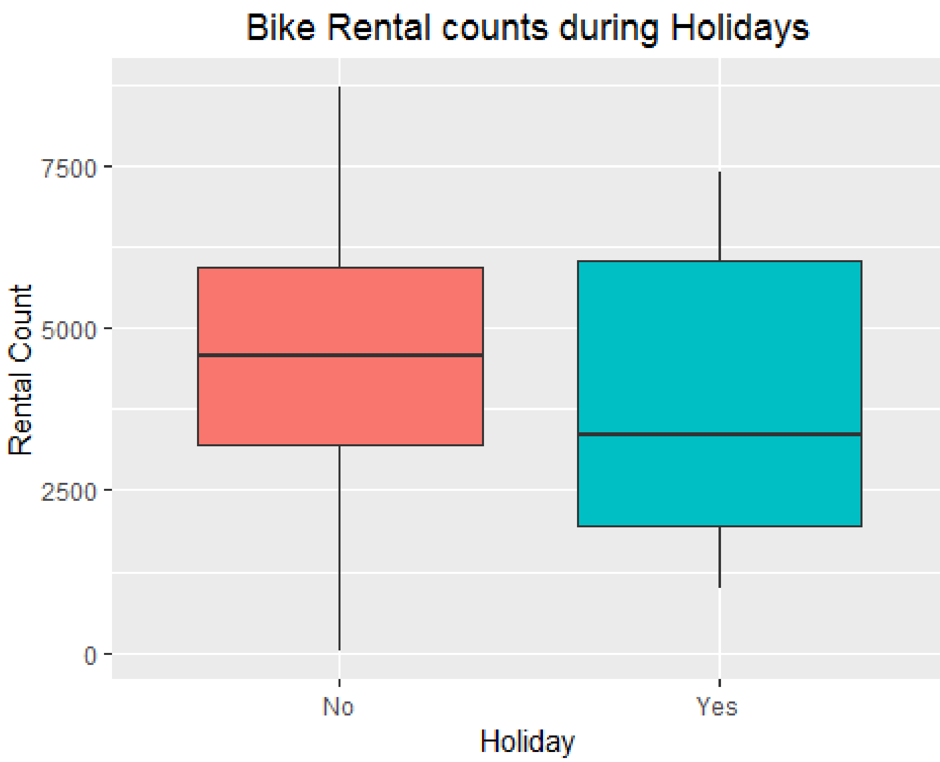





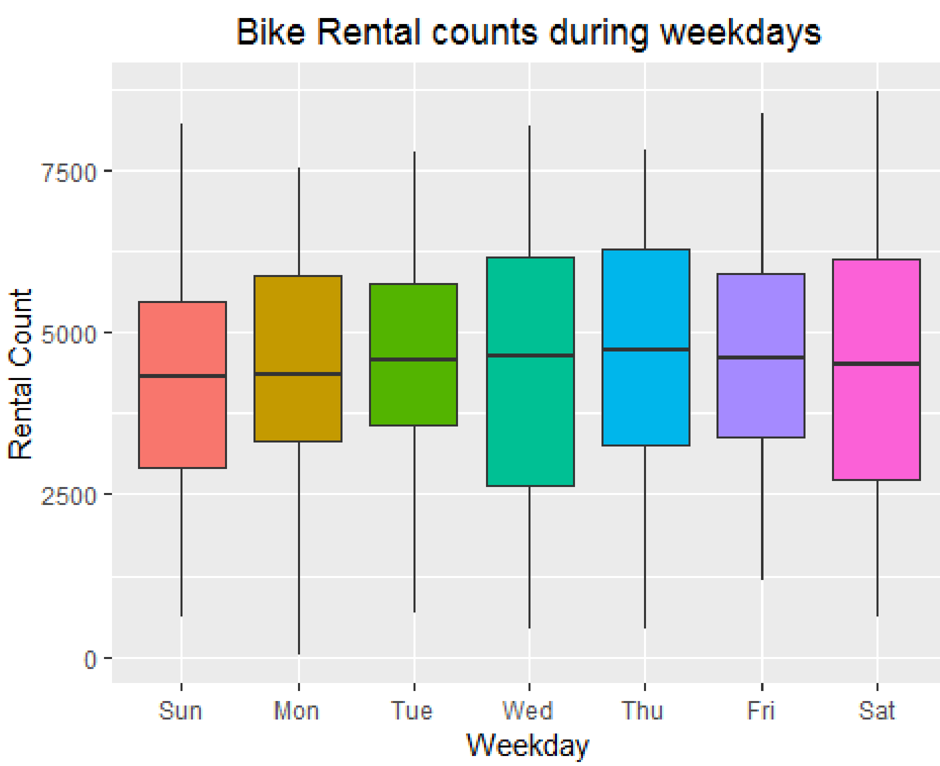
Month wise Bike Rental counts
Rental Count
0
2500
5000
7500
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month

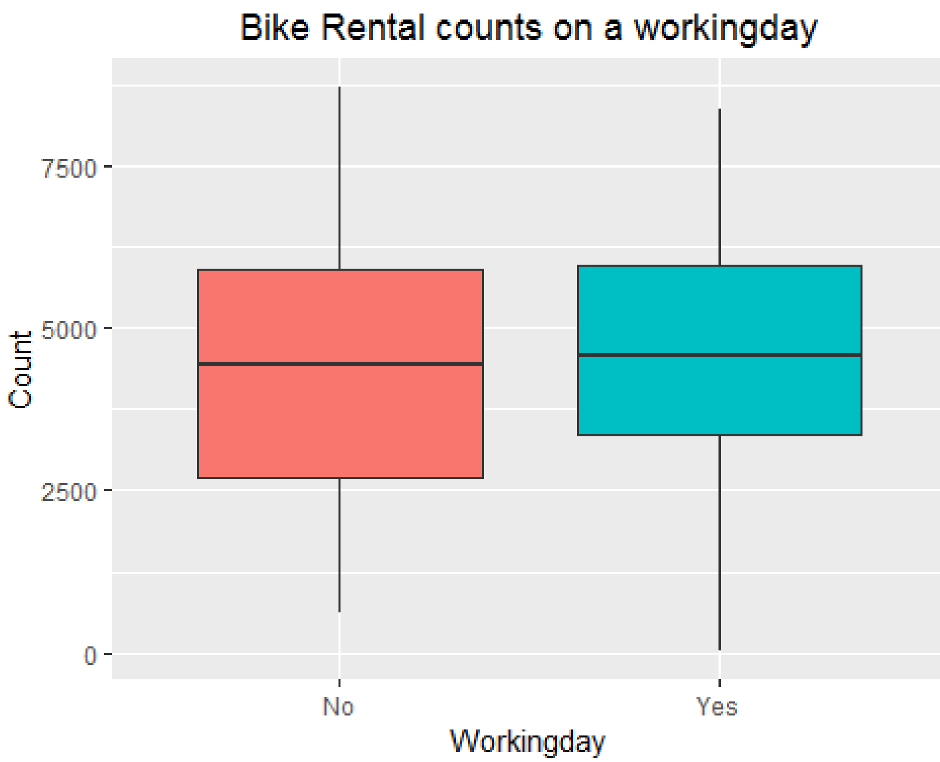
Day wise Bike Rental counts
Rental Count
0
2500
5000
7500
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Day

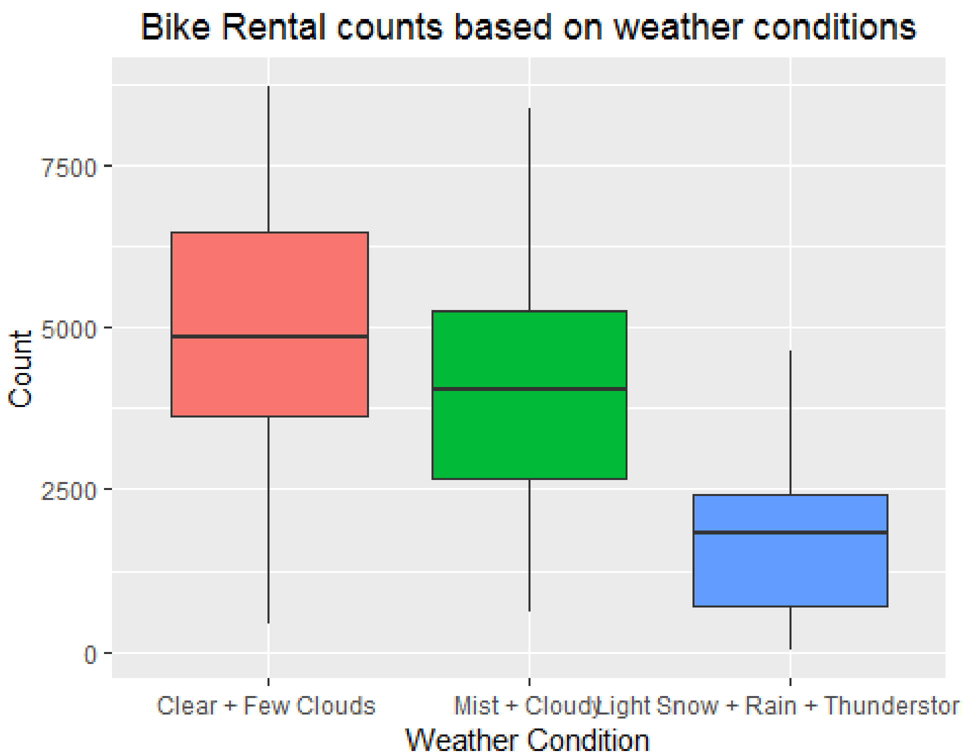


Bike Rental counts during Holidays
Rental Count
7500
5000
2500
0
No
Yes
Holiday

Bike Rental counts during weekdays
Rental Count
7500
5000
2500
0
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Weekday


Bike Rental counts on a workingday
7500
5000
2500
0
Count
No
Yes
Workingday


Bike Rental counts based on weather conditions
7500
5000
2500
0
Count
Clear + Few Clouds
Mist + Cloudy
Light Snow + Rain + Thunderstor
Weather Condition

From the above figures we can note the variation in rental counts for different seasons, month, day, year, weather conditions, weekdays and holidays. There were high rentals when the weather was clear with few clouds and low rentals during rain and thunderstorms. The rentals during 2012 was higher than 2011. We can also see the variations during different seasons will high rentals during fall.

### Distribution of Continuous variables

```
gridExtra::grid.arrange(Hg1, Hg2,Hg3,Hg4,Hg5,Hg6)
```

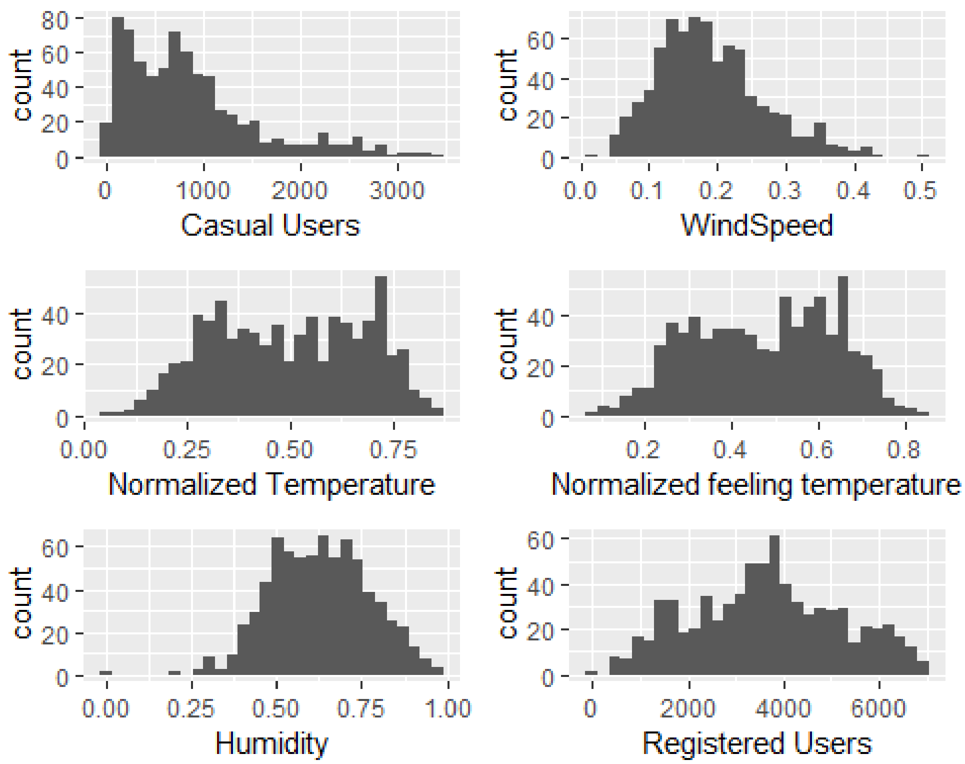



Figure 1.1 Distribution of continuous variables

From above figure, we see that most of the features are normally distributed. The Casual users is rightly skewed with some outliers which we will analyze in below steps.

## 2.1.2 Missing Value Analysis

Dealing with missing values is one of the first data preprocessing steps. They need to be treated by imputation methods or by ignoring the entire feature if the percentage of missing values are high. There are different methods of imputation like mean, median, mode, KNN method. In our dataset, we do not have any missing values as we can see from below data frame.

```
Missing_val = data.frame(apply(data_df, 2, function(x){sum(is.na(x))}))
Missing_val$Columns = row.names(Missing_val)
row.names(Missing_val) = NULL
names(Missing_val)[1] = "Count"
```

```
#Display Missing Value counts - There are no missing values
Missing_val = Missing_val[c(2,1)]
Missing_val
```

```
##        Columns Count
## 1      instant     0
## 2       dteday     0
## 3       season     0
## 4           yr     0
## 5         mnth     0
## 6      holiday     0
## 7      weekday     0
## 8   workingday     0
## 9   weathersit     0
## 10        temp     0
## 11       atemp     0
## 12         hum     0
## 13   windspeed     0
## 14      casual     0
## 15  registered     0
## 16         cnt     0
## 17         day     0
```

### 2.1.3 Outlier Analysis

Outlier Analysis is a part of data preprocessing steps. The outliers in the data needs to be dealt with as they can lead to misleading results in our prediction. Different ways to deal with outliers are by removing the variables if they have a higher percentage of outliers or by imputation method. We can use boxplot.stats to analyze the possible outliers in the feature and then deal with them accordingly. From below fig 1.2, we can see the outliers on a few variables with casual users feature having high percentage of outliers.

```
numeric_data = data_df[,c("windspeed","casual","hum")]
cnames = colnames(numeric_data)
for(i in 1:length(cnames)){
  assign(paste0("Gn",i),
         ggplot(data_df, aes_string(y = cnames[i]))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.color = "red", outlier.shape = 18, outlier.
size = 2))
}

gridExtra::grid.arrange(Gn1,Gn2,Gn3,ncol=3)
```

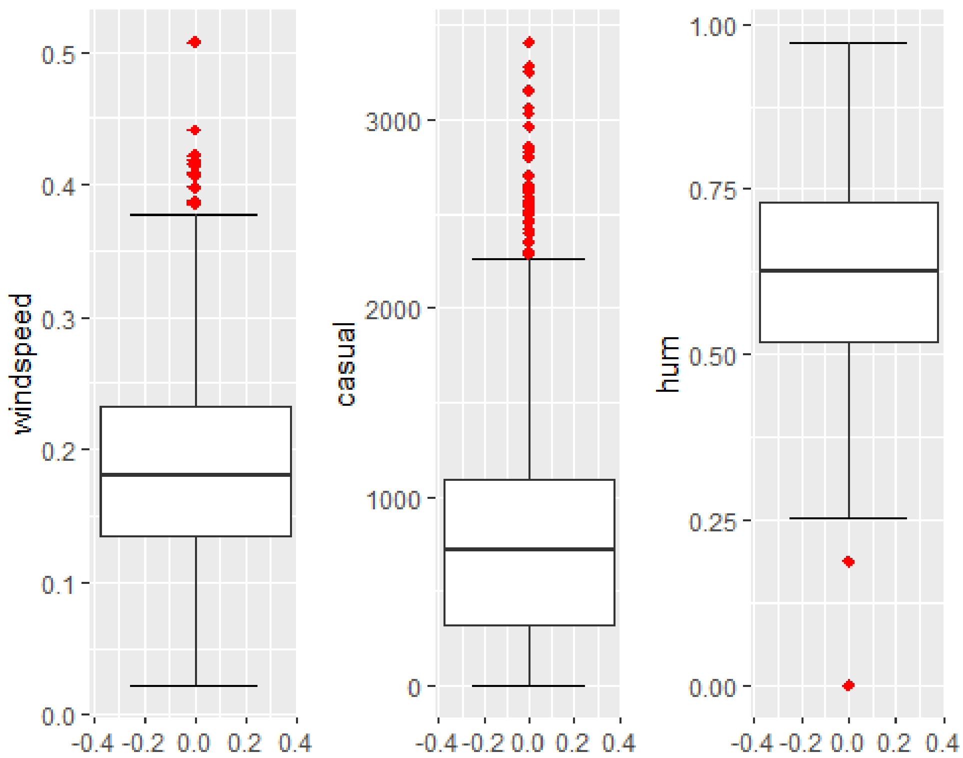



Figure 1.2 Outlier Analysis using boxplot

### 2.1.4 Feature Selection

Correlation analysis will help us determine the relation between two or more variables. If we have features that are high correlated, we can consider only one of the features instead of taking all the correlated features as they produce redundant information and can cause misleading results.

From our analysis and figure 1.3, we can see that the features temp and atemp has high correlation which needs to be dealt with for better performance of our model. There are many ways in dealing with multicollinearity which we will discuss below while dealing with multicollinearity cases.

```
#Correlation Analysis
  numeric_index = sapply(data_df, is.numeric)
  numeric_data = data_df[,numeric_index]
  train_cor = cor(numeric_data)
  corrplot(train_cor, method = 'color', addCoef.col = 'black')
```

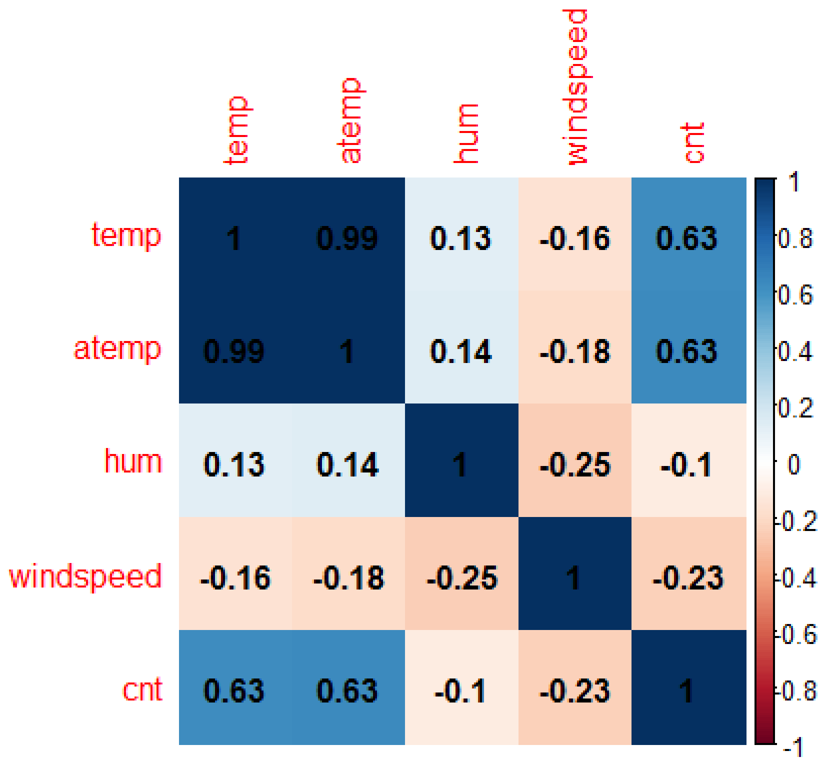



Figure 1.3 Correlation analysis

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that some of the features in our dataset do not add any information for our modeling. We need to get rid of those and some other features may have high correlation which can cause redundant information. So we need to deal with those as well by picking any one.

By using Random Forest algorithm we can categorize variables with high importance and consider them for our data modeling. From below figure 1.4, we see that feature yr has high importance compared to other features.

```
  #Feature selection using Random Forest

  RF_model = randomForest(cnt ~., data = data_df, ntree = 100, importance = T
RUE)
  pd = as.data.frame(importance(RF_model, type = 1))

  #Plot features with imporatance from High to Low
  ggplot(pd, aes(x = reorder(columns, -pd$`%IncMSE`), y = pd$`%IncMSE`, fill
= 'blue'))+
    geom_bar(stat = 'identity', show.legend = FALSE)+
    xlab("Variable Names")+
    ylab("Feature Importance Rate (%IncMSE)")+
    ggtitle("Features with Importance value")+
    theme(plot.title = element_text(hjust=0.5))
```

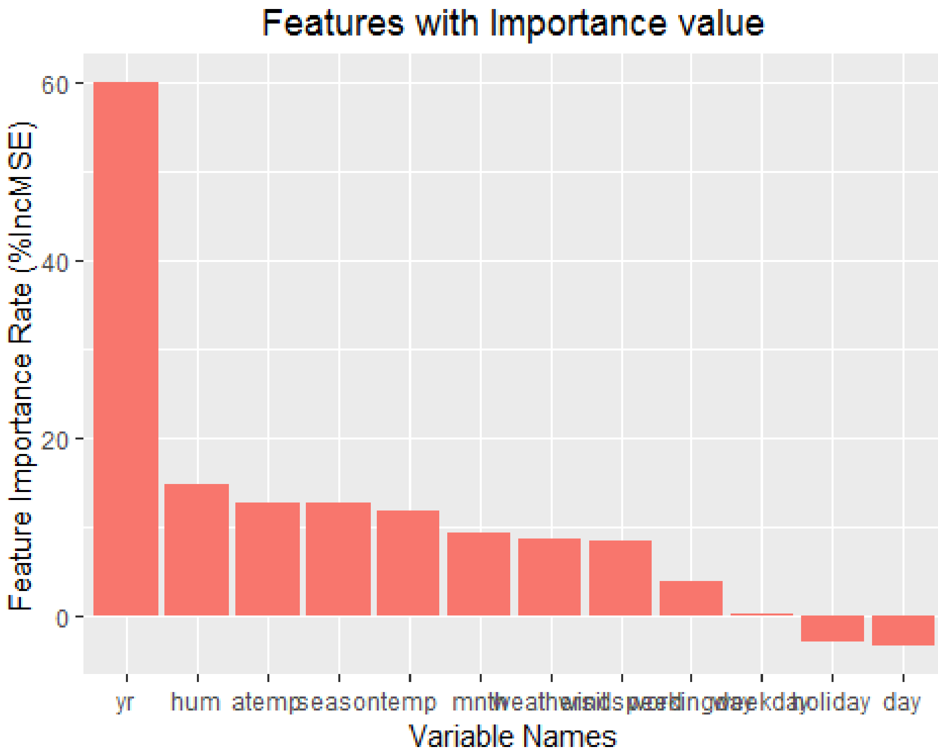



Figure 1.4 Features with Importance rate

## 2.2 Model Development

### 2.2.1 Data Splitting

We split the data into train dataset and evaluation or test data. The purpose is to apply or train the ML algorithms on the train dataset and to evaluate the model performance on our test dataset. The best model can be evaluated based on the accuracy of the prediction on the test dataset.

Here we have split the dataset into train and test using caret package.

```
  set.seed(1234)
  train_index = createDataPartition(data_df$cnt, p = 0.8, list = F)
  train = data_df[train_index,]
  test = data_df[-train_index,]

  dim(train)

## [1] 587  13

  dim(test)

## [1] 144  13
```

### 2.2.2 Multicollinearity

Multicollinearity occurs when two or more independent variables are highly correlated with each other which can lead to misleading results in building our model.

We can test the multicollinearity in our data using VIF(Variance Inflation Factor). Higher the value greater the variance. From below results using vifcor function we see that temp and atemp features have collinearity problem.

```
  vif(numeric_data[,-5])

##   Variables       VIF
## 1      temp 62.969819
## 2     atemp 63.632351
## 3       hum  1.079267
## 4 windspeed  1.126768

  vifcor(numeric_data[,-5], th = 0.9)

## 1 variables from the 4 input variables have collinearity problem:
##
## atemp
##
## After excluding the collinear variables, the linear correlation coefficien
ts ranges between:
## min correlation ( hum ~ temp ):  0.1269629
## max correlation ( windspeed ~ hum ):  -0.2484891
##
## ---------- VIFs of the remained variables --------
##   Variables      VIF
## 1      temp 1.034283
## 2       hum 1.074850
## 3 windspeed 1.084580
```

### 2.2.3 Multiple Linear Regression

Linear Regression is a modelling approach to find relationship between the dependent variable and one or more independent variables. After applying linear regression algorithm on all features, we got a R-Squared value of 84%. We can further try to improve our model efficiency by performing some regularization techniques or stepwise model selection.

```
  lm_model = lm(data = train, cnt~ .)
  summary(lm_model)

##
## Call:
## lm(formula = cnt ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3522.7  -370.8    54.3   458.9  2458.3
```

```
##
## Coefficients: (1 not defined because of singularities)
##                                               Estimate Std. Error t value
## (Intercept)                                   1216.587    342.182   3.555
## seasonSummer                                   946.415    207.978   4.551
## seasonFall                                     784.411    236.093   3.322
## seasonWinter                                  1508.130    201.190   7.496
## yr2012                                        1987.106     65.006  30.568
## mnthFeb                                         69.421    160.543   0.432
## mnthMar                                        516.221    195.772   2.637
## mnthApr                                        386.944    288.709   1.340
## mnthMay                                        699.186    313.272   2.232
## mnthJun                                        414.511    325.024   1.275
## mnthJul                                        -62.437    352.524  -0.177
## mnthAug                                        522.799    340.018   1.538
## mnthSep                                        995.448    294.087   3.385
## mnthOct                                        484.209    268.234   1.805
## mnthNov                                        -99.993    254.062  -0.394
## mnthDec                                        -90.872    200.966  -0.452
## holidayYes                                    -671.729    197.743  -3.397
## weekdayMon                                     291.928    123.775   2.359
## weekdayTue                                     307.003    121.395   2.529
## weekdayWed                                     396.235    121.384   3.264
## weekdayThu                                     384.152    119.948   3.203
## weekdayFri                                     428.101    121.388   3.527
## weekdaySat                                     566.295    124.602   4.545
## workingdayYes                                       NA         NA      NA
## weathersitMist + Cloudy                       -488.572     90.387  -5.405
## weathersitLight Snow + Rain + Thunderstorm -1767.697    233.122  -7.583
## temp                                          2550.835   1481.166   1.722
## atemp                                         2222.353   1531.693   1.451
## hum                                          -1546.873    355.115  -4.356
## windspeed                                    -2677.376    475.500  -5.631
## day 2                                          275.686    255.149   1.080
## day 3                                          282.892    253.341   1.117
## day 4                                          387.068    262.637   1.474
## day 5                                          193.747    272.175   0.712
## day 6                                          318.992    271.206   1.176
## day 7                                           50.645    256.180   0.198
## day 8                                           -7.757    268.307  -0.029
## day 9                                          178.327    264.293   0.675
## day10                                          376.196    253.065   1.487
## day11                                          461.379    255.882   1.803
## day12                                          305.921    256.628   1.192
## day13                                          263.576    265.544   0.993
## day14                                          435.662    272.958   1.596
## day15                                          446.758    257.710   1.734
## day16                                          422.559    256.354   1.648
## day17                                          722.665    265.811   2.719
## day18                                          111.797    255.922   0.437
```

```
## day19                                               298.371    255.445   1.168
## day20                                               398.096    260.916   1.526
## day21                                               355.254    257.038   1.382
## day22                                              -242.033    263.296  -0.919
## day23                                               116.829    264.991   0.441
## day24                                                51.891    255.657   0.203
## day25                                               -65.460    269.085  -0.243
## day26                                               285.966    262.178   1.091
## day27                                               160.820    272.594   0.590
## day28                                                98.299    259.227   0.379
## day29                                              -175.132    257.214  -0.681
## day30                                               -48.935    263.220  -0.186
## day31                                               552.304    303.343   1.821
##                                              Pr(>|t|)    
## (Intercept)                                  0.000411 ***
## seasonSummer                                 6.65e-06 ***
## seasonFall                                   0.000954 ***
## seasonWinter                                 2.80e-13 ***
## yr2012                                        < 2e-16 ***
## mnthFeb                                      0.665617    
## mnthMar                                      0.008615 ** 
## mnthApr                                      0.180739    
## mnthMay                                      0.026042 *  
## mnthJun                                      0.202755    
## mnthJul                                      0.859487    
## mnthAug                                      0.124754    
## mnthSep                                      0.000765 ***
## mnthOct                                      0.071617 .  
## mnthNov                                      0.694053    
## mnthDec                                      0.651328    
## holidayYes                                   0.000733 ***
## weekdayMon                                   0.018711 *  
## weekdayTue                                   0.011730 *  
## weekdayWed                                   0.001168 ** 
## weekdayThu                                   0.001444 ** 
## weekdayFri                                   0.000457 ***
## weekdaySat                                   6.82e-06 ***
## workingdayYes                                      NA    
## weathersitMist + Cloudy                      9.81e-08 ***
## weathersitLight Snow + Rain + Thunderstorm 1.53e-13 ***
## temp                                         0.085622 .  
## atemp                                        0.147398    
## hum                                          1.59e-05 ***
## windspeed                                    2.92e-08 ***
## day 2                                        0.280417    
## day 3                                        0.264655    
## day 4                                        0.141137    
## day 5                                        0.476873    
## day 6                                        0.240046    
## day 7                                        0.843362    
```

```
## day 8                                              0.976946
## day 9                                              0.500141
## day10                                              0.137729
## day11                                              0.071944 .
## day12                                              0.233764
## day13                                              0.321366
## day14                                              0.111070
## day15                                              0.083579 .
## day16                                              0.099878 .
## day17                                              0.006769 **
## day18                                              0.662405
## day19                                              0.243317
## day20                                              0.127667
## day21                                              0.167522
## day22                                              0.358387
## day23                                              0.659480
## day24                                              0.839236
## day25                                              0.807891
## day26                                              0.275888
## day27                                              0.555469
## day28                                              0.704690
## day29                                              0.496245
## day30                                              0.852589
## day31                                              0.069215 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 762.9 on 528 degrees of freedom
## Multiple R-squared:  0.8608, Adjusted R-squared:  0.8455
## F-statistic:  56.3 on 58 and 528 DF,  p-value: < 2.2e-16
```

```
  #Since we have multicollinearity issue between temp and atemp features we w
ill regularize it
  #Feature modeling using Stepwise modeling selection - both forward and back
ward
  lm_AIC = stepAIC(lm_model, direction = 'both')
```

```
## Start:  AIC=7847.74
## cnt ~ season + yr + mnth + holiday + weekday + workingday + weathersit +
##     temp + atemp + hum + windspeed + day
##
##
## Step:  AIC=7847.74
## cnt ~ season + yr + mnth + holiday + weekday + weathersit + temp +
##     atemp + hum + windspeed + day
##
##              Df Sum of Sq       RSS    AIC
## - day        30  25287408 332555269 7834.2
## <none>                    307267862 7847.7
## - atemp       1   1225084 308492946 7848.1
```

```
## - temp        1   1725999 308993861 7849.0
## - holiday     1   6715362 313983224 7858.4
## - weekday     6  13907175 321175036 7861.7
## - hum         1  11042135 318309997 7866.5
## - windspeed   1  18450198 325718060 7880.0
## - weathersit  2  37595239 344863101 7911.5
## - season      3  39495966 346763828 7912.7
## - mnth       11  50463459 357731321 7915.0
## - yr          1 543772441 851040302 8443.7
##
## Step:  AIC=7834.16
## cnt ~ season + yr + mnth + holiday + weekday + weathersit + temp +
##     atemp + hum + windspeed
##
##              Df Sum of Sq       RSS    AIC
## - atemp       1    776074 333331343 7833.5
## <none>                    332555269 7834.2
## - temp        1   2836961 335392230 7837.1
## - holiday     1   6575627 339130897 7843.7
## - weekday     6  13992851 346548120 7846.4
## + day        30  25287408 307267862 7847.7
## - hum         1  14628101 347183371 7857.4
## - windspeed   1  19729030 352284299 7866.0
## - weathersit  2  36002002 368557271 7890.5
## - mnth       11  49532534 382087803 7893.7
## - season      3  40576712 373131981 7895.7
## - yr          1 560334204 892889473 8411.9
##
## Step:  AIC=7833.53
## cnt ~ season + yr + mnth + holiday + weekday + weathersit + temp +
##     hum + windspeed
##
##              Df Sum of Sq       RSS    AIC
## <none>                    333331343 7833.5
## + atemp       1    776074 332555269 7834.2
## - holiday     1   6915190 340246533 7843.6
## - weekday     6  13933002 347264345 7845.6
## + day        30  24838397 308492946 7848.1
## - hum         1  14340767 347672110 7856.3
## - windspeed   1  22207712 355539055 7869.4
## - weathersit  2  37020589 370351932 7891.3
## - mnth       11  49080954 382412297 7892.2
## - season      3  40864355 374195698 7895.4
## - temp        1  64311871 397643214 7935.1
## - yr          1 559735938 893067281 8410.0
```

Stepwise model regression is a method of fitting regression models in which the choice of selecting predictor variables is carried by an automatic procedure. The model is evaluated based on AIC score. Lesser the score better the prediction rate of those variables considered.

In our analysis, we have used stepwise model selection on our linear regression model with both forward and backward approach which evaluates the model by iterating through each of the features and considering the best fit with score of AIC.

### 2.2.4 Decision Tree

Now, we will use a different regression model to predict our rental counts. We will use decision tree algorithm to predict the values of our target variable.

```
dt_model = rpart(cnt ~., data = train, method = 'anova')
dt_model
```

```
## n= 587
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 587 2207581000 4505.470
##    2) temp< 0.41875 227  497753300 2973.511
##      4) yr=2011 119  119353800 2124.571
##        8) season=Spring,Summer 83   25066270 1635.024 *
##        9) season=Winter 36   28535080 3253.250 *
##      5) yr=2012 108  198138100 3908.917
##       10) season=Spring 60   57924380 3152.817
##         20) atemp< 0.294789 33   21642160 2591.667 *
##         21) atemp>=0.294789 27   13190340 3838.667 *
##       11) season=Summer,Winter 48   63035970 4854.042
##         22) day= 9,22,23,24,25,27,30 9   13613860 3133.444 *
##         23) day= 1, 2, 3, 5, 6, 7, 8,10,11,12,13,14,15,16,17,18,19,20,21,2
6,28,29,31 39   16629380 5251.103 *
##    3) temp>=0.41875 360  841155500 5471.456
##      6) yr=2011 175  123368700 4269.377
##       12) mnth=Feb,Mar,Apr,Nov,Dec 35   20214490 3403.800 *
##       13) mnth=May,Jun,Jul,Aug,Sep,Oct 140   70375630 4485.771
##         26) hum>=0.886187 10    8787794 2938.700 *
##         27) hum< 0.886187 130   35812440 4604.777 *
##      7) yr=2012 185  225708300 6608.557
##       14) hum>=0.8095835 13   28133180 4633.308 *
##       15) hum< 0.8095835 172  143020700 6757.849
##         30) mnth=Jan,Mar,Apr,May,Jun,Jul,Nov,Dec 109   89141900 6476.523 *
##         31) mnth=Aug,Sep,Oct 63   30326460 7244.587 *
```

### 2.2.5 Random Forest

Random Forest is another algorithm that can be used for regression analysis. We will use randomForest on our train data set to predict our target variable.

```
rf_model = randomForest(cnt ~.,data = train, ntree = 100)
rf_model
```

```
##
## Call:
##  randomForest(formula = cnt ~ ., data = train, ntree = 100)
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 646323.1
##                     % Var explained: 82.81
```

# Chapter 3

## Conclusion

### 3.1 Model Evaluation

#### 3.1.1 Mean Absolute Percentage Error (MAPE)

MAPE or Mean Absolute Percentage Error is a measure of prediction accuracy. It is commonly used as a loss function for regression problems and in model evaluation. We will use this metric on our data models to evaluate the measure of prediction accuracy. MAPE can be calculated by taking the mean of (Actual value – Predicted value).

From below analysis, after predicting our built models on the test dataset, we then calculated the MAPE result. We see that linear regression algorithm with stepwise fitting regression model gave low prediction error of 19.93% and accuracy of 80.07%.

```
  predict_lm = predict(lm_model, test[,c(-12)])

## Warning in predict.lm(lm_model, test[, c(-12)]): prediction from a rank-
## deficient fit may be misleading

  predict_lm_AIC = predict(lm_AIC, test[,-12])
  predict_dt = predict(dt_model, test[,-12])
  predict_rf = predict(rf_model, test[,-12])

  #Finding error metrics
  #Calculating MAPE - y= Actual value, yhat = Predicted values
  mape = function(y,yhat){
    mean(abs((y-yhat)/y))*100
  }

  mape(test[,12], predict_lm)

## [1] 20.16607

  mape(test[,12], predict_lm_AIC)

## [1] 19.93934

  mape(test[,12],predict_dt)

## [1] 26.98542

  mape(test[,12], predict_rf)

## [1] 24.413
```

### 3.1.2 Mean Squared Error (MSE)

Mean Squared Error(MSE) and Root Mean Squared Error (RMSE) can also be used as our metrics to evaluate the performance of our models. We can use regr.eval() function to get our RMSE, MSE, MAPE and MAE results. Based on these metrics we can evaluate which model gives good performance and better accuracy.

Based on our below evaluation, we see that predict_lm_AIC model gives better MSE results.

```
regr.eval(test[,12],predict_lm, stats = c("rmse","mse","mape","mae"))

##         rmse          mse         mape          mae
## 8.100396e+02 6.561641e+05 2.016607e-01 6.103268e+02

regr.eval(test[,12],predict_lm_AIC, stats = c("rmse","mse","mape","mae"))

##         rmse          mse         mape          mae
## 7.823221e+02 6.120279e+05 1.993934e-01 5.920271e+02

regr.eval(test[,12],predict_dt, stats = c("rmse","mse","mape","mae"))

##         rmse          mse         mape          mae
## 1.005612e+03 1.011255e+06 2.698542e-01 7.234144e+02

regr.eval(test[,12],predict_rf, stats = c("rmse","mse","mape","mae"))

##         rmse          mse         mape          mae
##    810.10792 656274.84090      0.24413    597.85694
```

## 3.2 Final Model Selection

From our model evaluation, we can see that stepwise fitting regression model on linear regression algorithm gives us low error rate and better results compared to decision trees and random forest.

# Appendix A - R Code

```
###################### Feature Engineering #############################
data_df$season = as.factor(data_df$season)
levels(data_df$season) = c("Spring","Summer","Fall","Winter")
data_df$yr = as.factor(data_df$yr)
levels(data_df$yr) = c("2011","2012")
data_df$mnth = as.factor(data_df$mnth)
levels(data_df$mnth) = c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep
","Oct","Nov","Dec")
data_df$holiday = as.factor(data_df$holiday)
levels(data_df$holiday) = c("No","Yes")
data_df$weekday = as.factor(data_df$weekday)
levels(data_df$weekday) = c("Sun","Mon","Tue","Wed","Thu","Fri","Sat")
data_df$workingday = as.factor(data_df$workingday)
levels(data_df$workingday) = c("No", "Yes")
data_df$weathersit = as.factor(data_df$weathersit)
levels(data_df$weathersit) = c("Clear + Few Clouds","Mist + Cloudy", "Light S
now + Rain + Thunderstorm")
data_df$dteday = as.POSIXct(data_df$dteday)
data_df$day = strftime(data_df$dteday, '%e')
data_df$day = as.factor(data_df$day)

################### Exploratory Data Analysis ############################
#Analysis on dependent variable - Count
ggplot(data = data_df, aes(cnt))+
  geom_histogram()
```

```
#Analysis of Independent variabls vs dependent variable
  #Season vs Rental counts
  g1 = ggplot(data = data_df, aes(data_df$season, data_df$cnt, fill = season)
)+
    geom_boxplot(show.legend = FALSE)+
    xlab("Season")+
    ylab("Rental Count")+
    labs(title = "Bike Rental counts during seasons")+
    theme(plot.title = element_text(hjust=0.5))

  g2 = ggplot(data = data_df, aes(data_df$yr, data_df$cnt, fill = data_df$yr)
)+
    geom_boxplot(show.legend = FALSE)+
    xlab("Year")+
    ylab("Rental Count")+
    labs(title = "Year wise Bike Rental counts")+
    theme(plot.title = element_text(hjust=0.5))
```

```
  g3 = ggplot(data = data_df, aes(data_df$mnth, data_df$cnt, fill = data_df$m
nth))+
    geom_boxplot(show.legend = FALSE)+
    xlab("Month")+
    ylab("Rental Count")+
    labs(title = "Month wise Bike Rental counts")+
    theme(plot.title = element_text(hjust=0.5))

  g4 = ggplot(data = data_df, aes(data_df$day, data_df$cnt, fill = data_df$da
y))+
    geom_boxplot(show.legend = FALSE)+
    xlab("Day")+
    ylab("Rental Count")+
    labs(title = "Day wise Bike Rental counts")+
    theme(plot.title = element_text(hjust=0.5))

  g5 = ggplot(data = data_df, aes(data_df$holiday, data_df$cnt, fill = data_d
f$holiday))+
    geom_boxplot(show.legend = FALSE)+
    xlab("Holiday")+
    ylab("Rental Count")+
    labs(title = "Bike Rental counts during Holidays")+
    theme(plot.title = element_text(hjust=0.5))

  g6 = ggplot(data = data_df, aes(data_df$weekday, data_df$cnt, fill = data_d
f$weekday))+
    geom_boxplot(show.legend = FALSE)+
    xlab("Weekday")+
    ylab("Rental Count")+
    labs(title = "Bike Rental counts during weekdays")+
    theme(plot.title = element_text(hjust=0.5))

  g7 = ggplot(data = data_df, aes(data_df$workingday, data_df$cnt, fill = dat
a_df$workingday))+
    geom_boxplot(show.legend = FALSE)+
    xlab("Workingday")+
    ylab("Count")+
    labs(title = "Bike Rental counts on a workingday")+
    theme(plot.title = element_text(hjust=0.5))

  g8 = ggplot(data = data_df, aes(data_df$weathersit, data_df$cnt, fill = dat
a_df$weathersit))+
    geom_boxplot(show.legend = FALSE)+
    xlab("Weather Condition")+
    ylab("Count")+
    labs(title = "Bike Rental counts based on weather conditions")+
    theme(plot.title = element_text(hjust=0.5))
```

```
  gridExtra::grid.arrange(g1,g2,g3,g4,g5,g6,g7,g8,ncol=2)
```

```
  #Distribution of Continuous variables
  Hg1 = ggplot(data_df, aes(x = casual))+
    geom_histogram()+
    xlab("Casual Users")

  Hg2 = ggplot(data_df, aes(x = windspeed))+
    geom_histogram()+
    xlab("WindSpeed")

  Hg3 = ggplot(data_df, aes(x = temp))+
    geom_histogram()+
    xlab("Normalized Temperature")

  Hg4 = ggplot(data_df, aes(x = atemp))+
    geom_histogram()+
    xlab("Normalized feeling temperature")

  Hg5 = ggplot(data_df, aes(x = hum))+
    geom_histogram()+
    xlab("Humidity")

  Hg6 = ggplot(data_df, aes(x = registered))+
    geom_histogram()+
    xlab("Registered Users")

  gridExtra::grid.arrange(Hg1, Hg2,Hg3,Hg4,Hg5,Hg6)

################## Missing Values Analysis #############################
  Missing_val = data.frame(apply(data_df, 2, function(x){sum(is.na(x))}))
  Missing_val$Columns = row.names(Missing_val)
  row.names(Missing_val) = NULL
  names(Missing_val)[1] = "Count"
  #Display Missing Value counts - There are no missing values
  Missing_val = Missing_val[c(2,1)]

########################### Outlier Analysis #####################
#Check for numeric variables and store them in numeric_index
  numeric_index = sapply(data_df, is.numeric)
  numeric_data = data_df[,numeric_index]

  boxplot.stats(numeric_data$temp)$out

## numeric(0)

  boxplot.stats(numeric_data$atemp)$out
```

```
## numeric(0)

  boxplot.stats(numeric_data$hum)$out

## [1] 0.187917 0.000000

  boxplot.stats(numeric_data$windspeed)$out

##  [1] 0.417908 0.507463 0.385571 0.388067 0.422275 0.415429 0.409212
##  [8] 0.421642 0.441563 0.414800 0.386821 0.398008 0.407346

  boxplot.stats(numeric_data$casual)$out

##  [1] 2355 2282 3065 2418 2521 2397 3155 2469 2301 2347 3252 2795 2846 2541
## [15] 2496 2622 3410 2704 2855 3283 2557 2795 2494 2708 2963 2634 2657 2551
## [29] 2562 2355 2544 2345 2827 2352 2613 2570 3160 2512 2454 2589 3031 2806
## [43] 2643 2290

  boxplot.stats(numeric_data$registered)$out

## integer(0)

  numeric_data = data_df[,c("windspeed","casual","hum")]
  cnames = colnames(numeric_data)
  for(i in 1:length(cnames)){
    assign(paste0("Gn",i),
           ggplot(data_df, aes_string(y = cnames[i]))+
             stat_boxplot(geom = "errorbar", width = 0.5) +
             geom_boxplot(outlier.color = "red", outlier.shape = 18, outlier.
size = 2))
  }

  gridExtra::grid.arrange(Gn1,Gn2,Gn3,ncol=3)
```

```
######################### Feature Selection ##############################
  #Remove Casual and Registered from the dataset as this sums up total counts
  #Remove instant variable and dteday as they do not give much information
  data_df = data_df[,-c(1,2,14,15)]

  #Correlation Analysis
  numeric_index = sapply(data_df, is.numeric)
  numeric_data = data_df[,numeric_index]

  train_cor = cor(numeric_data)
  corrplot(train_cor, method = 'color', addCoef.col = 'black')
```

```
  #Feature selection through Random Forest
  RF_model = randomForest(cnt ~., data = data_df, ntree = 100, importance = T
```

```
RUE)
  pd = as.data.frame(importance(RF_model, type = 1))
  pd$columns = row.names(pd)
  row.names(pd) = NULL
  pd = pd[order(-pd$`%IncMSE`),]
  pd = pd[,c(2,1)]
  View(pd)

  #Plot features with imporatance from High to Low
  ggplot(pd, aes(x = reorder(columns, -pd$`%IncMSE`), y = pd$`%IncMSE`, fill
= 'blue'))+
    geom_bar(stat = 'identity', show.legend = FALSE)+
    xlab("Variable Names")+
    ylab("Feature Importance Rate (%IncMSE)")+
    ggtitle("Features with Importance value")+
    theme(plot.title = element_text(hjust=0.5))



################# ModelDevelopment#####################################
  #Split data
  set.seed(1234)
  train_index = createDataPartition(data_df$cnt, p = 0.8, list = F)
  train = data_df[train_index,]
  test = data_df[-train_index,]

  #Check for multicollinearity
  #library(usdm)
  vif(numeric_data[,-5])

  vifcor(numeric_data[,-5], th = 0.9)

  #Multicollinearity issues are found between temp and atemp features

  ################# Linear Regression Modelling ################
  lm_model = lm(data = train, cnt~ .)
  summary(lm_model)

  #Finding error metrics
  #Calculating MAPE - y= Actual value, yhat = Predicted values
  mape = function(y,yhat){
    mean(abs((y-yhat)/y))*100
  }

  mape(test[,12], predict_lm)

  #Alternate method - regr.eval() from DMwR library gives completed evaluatio
n of regression model
  regr.eval(test[,12],predict_lm, stats = c("rmse","mse","mape","mae"))
```

```r
  #Since we have multicollinearity issue between temp and atemp features we w
ill regularize it
  #Feature modeling using Stepwise modeling selection - both forward and back
ward
  lm_AIC = stepAIC(lm_model, direction = 'both')

  predict_lm_AIC = predict(lm_AIC, test[,-12])
  summary(predict_lm_AIC)

  mape(test[,12], predict_lm_AIC)

  regr.eval(test[,12],predict_lm_AIC, stats = c("rmse","mse","mape","mae"))

  ######################### Decision Tree #########################
  dt_model = rpart(cnt ~., data = train, method = 'anova')

  predict_dt = predict(dt_model, test[,-12])
  summary(predict_dt)

  mape(test[,12],predict_dt)

  regr.eval(test[,12],predict_dt, stats = c("rmse","mse","mape","mae"))

 ######################### Random Forest #########################
  set.seed(3210)
  rf_model = randomForest(cnt ~.,data = train, ntree = 100)

  predict_rf = predict(rf_model, test[,-12])
  summary(predict_rf)

  mape(test[,12], predict_rf)
  regr.eval(test[,12],predict_rf, stats = c("rmse","mse","mape","mae"))
```