Bike Rental Count Prediction-R

Chaitrali A Deokar

April 2019

Contents

Chapter 1

INTRODUCTION

1.1 Problem Statement

Bike rental count per day varies greatly based on the environmental settings like weather(sunny or raining) , whether it is a working day or holiday, etc. The objective of this analysis is to Predication of bike rental count on daily based on the environmental and seasonal settings. This will help the renting agencies identify the demand (increase or decrease) for a particular day and ensure adequate supply of bikes to optimise porift.

1.2 Data

The Analysis involves building a Regression model to predict the Bike rental count depending on a number of input parameters. The given data is a CSV file that consists 16 variables and 731 Observation. Given below is a sample of the data that we develop our model on

Instant	Dteday	season	yr	mnth	holiday	weekday	Workingday	weathersit
1	01-01-2011	1	0	1	0	6	0	2
2	02-01-2011	1	0	1	0	0	0	2
3	03-01-2011	1	0	1	0	1	1	1
4	04-01-2011	1	0	1	0	2	1	1
5	05-01-2011	1	0	1	0	3	1	1

Table 1.1 Columns 1- 9 of the "day.csv" data

weathersit	Temp	Atemp	Hum	windspeed	Casual	Registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

Table 1.2 Columns 9-16 of the "day.csv" data

The details of data attributes in the dataset are as follows –

1. instant: Record index
2. dteday: Date
3. season: Season (1:springer, 2:summer, 3:fall, 4:winter)
4. yr: Year (0: 2011, 1:2012)
5. mnth: Month (1 to 12)
6. holiday: weather day is holiday or not (extracted fromHoliday Schedule)
7. weekday: Day of the week
8. workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
9. weathersit: (extracted fromFreemeteo)
    i. 1: Clear, Few clouds, Partly cloudy, Partly cloudy
    ii. 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

    iii. 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    iv. 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

10. temp: Normalized temperature in Celsius. The values are derived via
    i. (t-t_min)/(t_max-t_min), t_min=-8,

    ii. t_max=+39 (only in hourly scale)
11. atemp: Normalized feeling temperature in Celsius. The values are derived via
    i. (t-t_min)/(t_maxt_min), t_min=-16,

    ii. t_max=+50 (only in hourly scale)
12. hum: Normalized humidity. The values are divided to 100 (max)
13. windspeed: Normalized wind speed. The values are divided to 67 (max)
14. casual: count of casual users
15. registered: count of registered users
16. cnt: count of total rental bikes including both casual and registered

From the above sample it is seen that the 2 to 13 are the the seasonal setting variables like temperature, the humidity etc on which the rental count per day is dependent. Hence, the are the predictor variables

The dependent variables are 3 namely, casual, registered and cnt. Among these our target variable is “cnt” which is basically a sum of the “casual” and “registered variables”

Hence below is a list of the dependent variables

Dependent Variables
Dteday
Season
Yr
Mnth
Holiday
Weekday
Workingday
Weathersit
Temp
Atemp
Hum
Windspeed

Table1.3 List of Dependent Variables

And, the dependent variables whose output depends on these are

Target variable
Casual
Registered
Cnt

Table 1.4 List of Dependent Variables

Chapter 2

Methodology

2.1 CRISP DM Process

CRISP-DM stands for cross-industry process for data mining. The CRISP-DM methodology provides a structured approach to planning a data mining project. The project follows CRISP Dm process to develop the model for the given problem. It involves the following steps:

1. Business understanding
2. Data understanding
3. Data preparation
4. Modeling
5. Evaluation
6. Deployment

2.2 Business Understanding

1. **Set objectives** - This primary objective from a business perspective would be " How much do the seasonal settings affect the Bike rental count?" and "to Predict the Bike rental count wit variation in the external seasonal settings".
2. **Produce project plan** – The plan will involve Understanding the data and converting into a proper shape , i.e., a data free from any missing values, outliers that can possibly produce errors; scaling the ; applying various models and choosing the best model to predict the bike rental count
3. **Business success criteria** – The success would be related to able to accurately predict the bike rental count for a particular input , with maximum accuracy . On being able to predict it accurately, sufficient bikes will be made available to ensure uninterrupted supply of bikes to meet the demand each day.

2.3 Data Understanding

The given data is a CSV file that consists 16 variables and 731 Observation.
The details of data attributes in the dataset are as follows –

1. instant: Record index
2. dteday: Date
3. season: Season (1:springer, 2:summer, 3:fall, 4:winter)
4. yr: Year (0: 2011, 1:2012)
5. mnth: Month (1 to 12)
6. holiday: weather day is holiday or not (extracted fromHoliday Schedule)
7. weekday: Day of the week
8. workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
9. weathersit: (extracted fromFreemeteo)
   i. 1: Clear, Few clouds, Partly cloudy, Partly cloudy
   ii. 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
   iii. 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
   iv. 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
10. temp: Normalized temperature in Celsius. The values are derived via
    i. (t-t_min)/(t_max-t_min), t_min=-8,
    ii. t_max=+39 (only in hourly scale)

11. atemp: Normalized feeling temperature in Celsius. The values are derived via
    i. (t-t_min)/(t_maxt_min), t_min=-16,

    ii. t_max=+50 (only in hourly scale)
12. hum: Normalized humidity. The values are divided to 100 (max)
13. windspeed: Normalized wind speed. The values are divided to 67 (max)
14. casual: count of casual users
15. registered: count of registered users
16. cnt: count of total rental bikes including both casual and registered

2.4 Data Pre Processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

2.4.1 Missing Value Analysis

Missing value is the occurance of incomplete observations in asset of data. Missing values can arise due to many reasons, error in uploading data, unable to measure a particular observation etc.

Due to presence of missing values, that observation either gets value as 0 or NA. These affect the accuracy of model. Hence it is necessary to check for any missing values in the given data.

Missing Value Analysis in R:

To check if there are any null vaues in R the following code is applied:

```
#check for missing values
missing_val = data.frame(apply(Rental_train,2,function(x){sum(is.na(x))}))
```

The output of this code shows that there are no missing values in the data.

Print(missing_val)

variable	missing percentage
dteday	0
season	0
yr	0
mnth	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0
hum	0
windspeed	0

Table 2.1 Missing Value Output

This shows that there are no missing values.

2.4.2 Outlier Analysis

An Outlier is any inconsistent or abnormal observation that deviates from the rest of the observations. These inconsistent observation can be due to manual error, poor quality/ malfunctioning equipments or correct but exceptional data. It can cause an error in prediciting the target variables.

Hence we need to check for the outliers and either remove the observations containing them or replace them with NA and later impute values.

We visualize the outliers using boxplots. We plot the continuous variables using Box plot and the following are the results:

Box Plot in R

```
#  BoxPlots - Distribution and Outlier Check
numeric_data = Rental_train[,nu]
#excluding the output quantities
cnames = colnames(numeric_data)[1:4]
cnames

for (i in 1:length(cnames))
{
 assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"), data = Rental_train)+
        stat_boxplot(geom = "errorbar", width = 0.5) +
        geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
               outlier.size=1, notch=FALSE) +
        theme(legend.position="bottom")+
        labs(y=cnames[i],x="y")+
        ggtitle(paste("Box plot of rental count for",cnames[i])))
}

## Plotting plots together
gridExtra::grid.arrange(gn1,gn2,ncol=2)
```

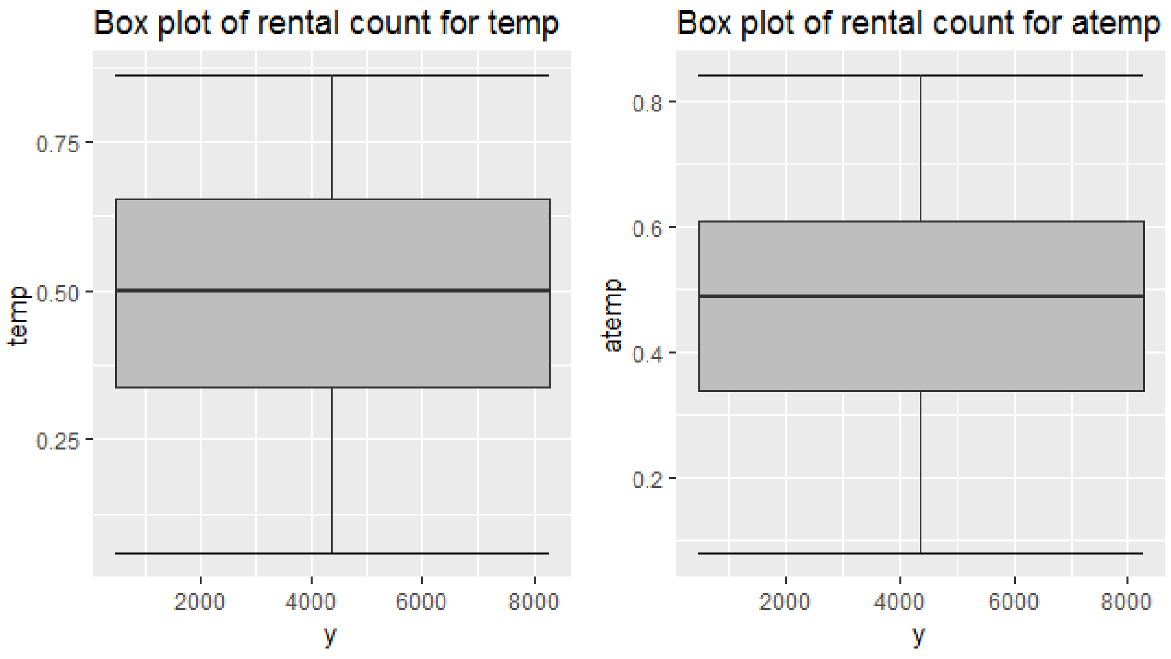

Fig 1 Box plot of temp and atemp

```
gridExtra::grid.arrange(gn3, gn4,ncol=2)
```

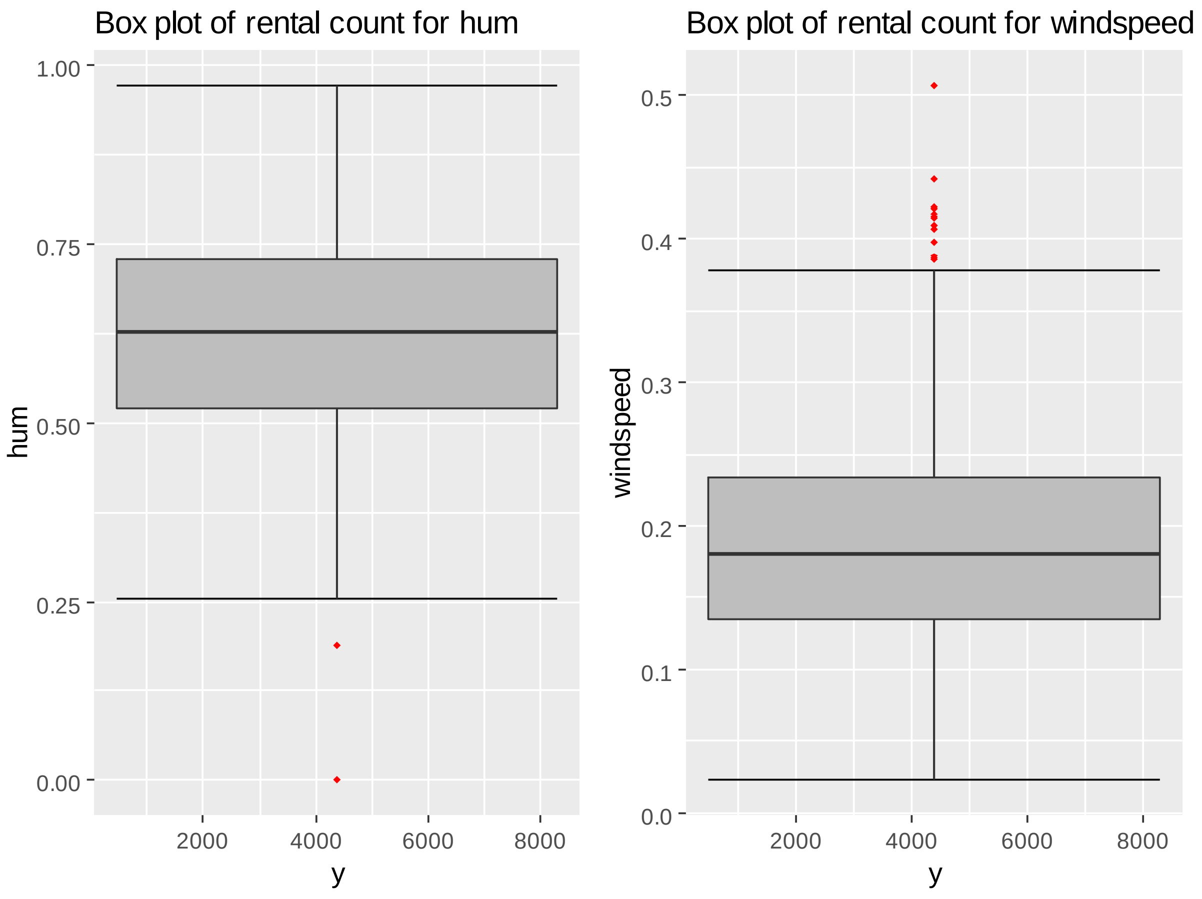

Fig 2 Box plot of temp and a temp

The figures Shows that there are no outliers in the temp, atemp variable whereas there are outliers in

the "hum" and "windspeed" variable

These outliers can be removed by 2 ways:

1. Removing the observations consisting of the Outliers
2. Replacing outliers with NA and the imputing values.

In our case, we chose to Replace the outliers with NA. The values can be imputed using 3 methods namely

1. Mean method
2. Median Method
3. KNN imputation method

We did a trial and the Mean method worked better, hence we imputed the NAs in Hum with the Mean value of the particular variable.

2.4.3 Feature Selection

Before creating a model we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of prediction. Hence, feature selection can help in reducing the time for computation of model as well as the complexity of the model. Also, few models require the independent variables to be free from multicollinear effect, hence it is needed to perform various procedures to ensure that the independent variables are not collinear.

The 1st step involved is to do the correlation analysis for the categorical variables and continuous variables. Since our Predictor variable is also continuous we will plot a correlation table that would predict the correlation strength between the dependent variable and the 'cnt', 'casual' and 'registered' variable

Correlation Analysis in R For Continuous variables

## Correlation Plot corrgram(Rental_train[,nu], order = F, upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

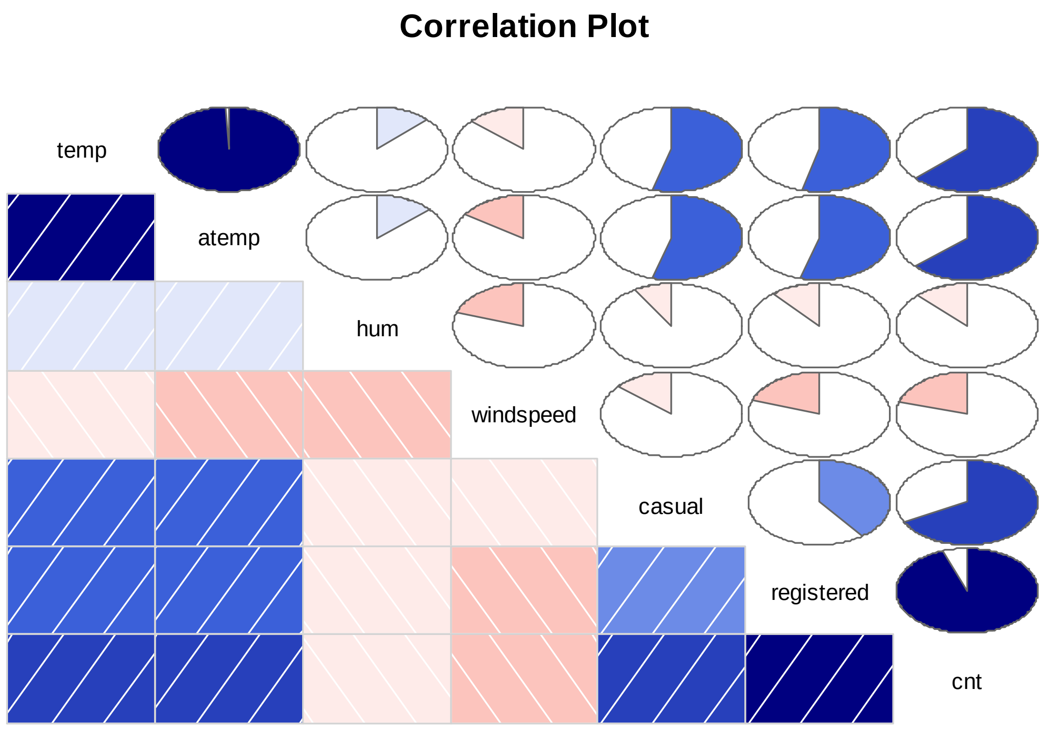

Fig 3 Correlation Plot for Continuous variables in R

The above plot shows that the "temp" and the "atemp" variables are the most correlated with the "casual", "registered" and "cnt" variable. Also, the "temp" and "atemp" variable are very closely related to each other.

To understand this in more quantitative terms, we plot the values below:

#Finding the correlation between the numeric variables

```
num_cor=round(cor(numeric_data), 3)
```

Var/Var	temp	atemp	hum	Windspeed	casual	registered	cnt
Temp	1	0.992	0.127	-0.158	0.543	0.54	0.627
Atemp	0.992	1	0.14	-0.184	0.544	0.544	0.631
Hum	0.127	0.14	1	-0.248	-0.077	-0.091	-0.101
windspeed	-0.158	-0.184	-0.248	1	-0.168	-0.217	-0.235
casual	0.543	0.544	-0.077	-0.168	1	0.395	0.673
Registered	0.54	0.544	-0.091	-0.217	0.395	1	0.946
Cnt	0.627	0.631	-0.101	-0.235	0.673	0.946	1

Table 2.2 Correlation Values for continuous Variables

Correlation Analysis in R For Categorical variables

For Categorrical data we do the Chi square test to check if a pair of categorical data are correlated or not.

```
# ## Chi-squared Test of Independence

factor_data = Rental_train[,fac]



for (i in 1:8){

 for (p in 1:8){

  # print(names(factor_data)[i])

  # print(names(factor_data)[p])

  print(chisq.test(table(factor_data[,p],factor_data[,i]), p-value))

  }

 }
```

The p value only give an idea if the values are dependent or not, but to check the strength of association, we use GoodMan Krussels test. It gives the following results.

```
#to check the degree of association of categorical variable

library(GoodmanKruskal)

plot(GKtauDataframe( factor_data), corrColors = 'blue')
```

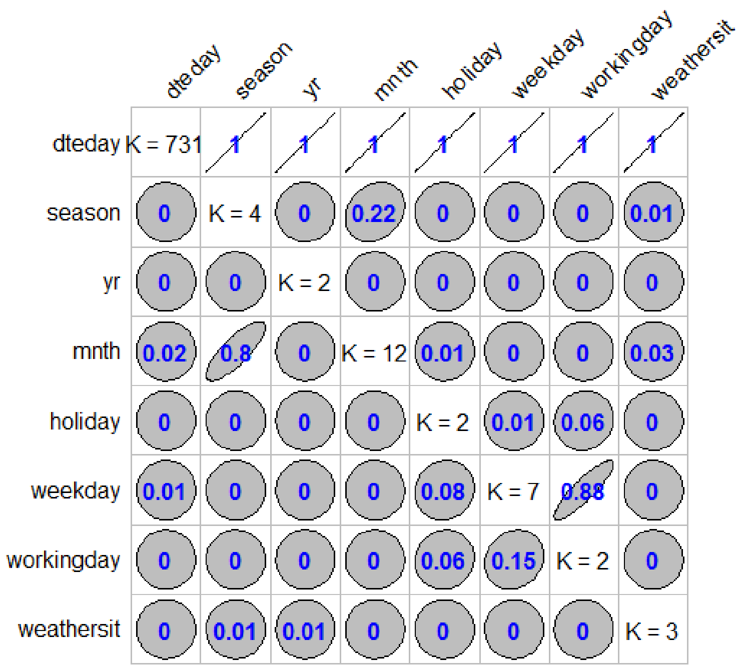

Fig 4 Strength of Correlation Among Categorical Variables

From the above R correlation, the following is seen:

1. Temp and atemp are highly related
2. Temp and atemp are highly correlated with the target variable "cnt"

Hence from the above we can choose to remove the following variables for our model development:

1. dteday- since they are just date instants
2. The "casual" variable, because it is not needed to be predicted as cnt is the target variable
3. The "Registeredl" variable, because it is not needed to be predicted as cnt is the target variable
4. Both temp and atemp are highly correlated, but since they are highly correlated with target variable, we do not remove them.

2.5 Modeling in R

We know that the dependent variabes are "cnt" , "casual" , "registered" . The "cnt" variable is a sum of the "casual" and "registered" variable. Hence we can choose to take only the "cnt" variable as the output to simplify the model. The model is to predict count, hence it is a **Regression** problem statement.

The various models that can be used for prediction problem statement are Decision trees, Random Forest, Linear Regression and KNN imputation.

2.5.1 Decision Tree

The Decision tree algorithm is applied on both R

- fit = rpart(cnt ~., data = train, method = "anova")
- predictions_DT = predict(fit, test[,-12])

```
> fit
n= 584

node), split, n, deviance, yval
      * denotes terminal node
```

```
1) root 584 2152412000 4480.755
  2) temp< 0.4420835 246  591562400 3081.354
    4) yr=0 128  132300000 2216.406
      8) season=1,2 89   28124960 1707.618 *
      9) season=4 39   28559770 3377.487 *
    5) yr=1 118  259624700 4019.602
     10) season=1 63   61148640 3186.492
       20) atemp< 0.260444 23   15108480 2393.739 *
       21) atemp>=0.260444 40   23274300 3642.325 *
     11) season=2,4 55  104662800 4973.891
       22) hum>=0.765417 10   31032900 3196.700 *
       23) hum< 0.765417 45   35027180 5368.822 *
  3) temp>=0.4420835 338  728480400 5499.254
    6) yr=0 167  111150800 4328.916
     12) hum>=0.849375 16   17901250 3151.938 *
     13) hum< 0.849375 151   68736520 4453.629 *
    7) yr=1 171  165203200 6642.216
     14) hum>=0.8322915 7    3959767 4662.857 *
     15) hum< 0.8322915 164  132647800 6726.701 *
```

The above shows the rules of splitting od threes. The main root spilts into 2 nodes having temp< 0.4420835 and temp>=0.4420835 as their conditions. Similarly the nodes further split. We apply these rules to predict the test cases.

2.5.2 Random Forest

```
>RF_model = randomForest(cnt ~. , train, importance = TRUE, ntree = 300)
>RF_Predictions = predict(RF_model, test[,-12])
```

```
> importance(RF_model, type = 1)
        %IncMSE
season     22.612920
yr        85.602701
mnth      16.398877
holiday    2.303832
weekday    1.643227
workingday  3.680979
weathersit 15.976428
temp      19.588411
atemp     20.846865
hum       23.331496
windspeed   8.241200
```

The above shows that the most important variable for predicting the rental count is yr and the least important is weekday.

2.5.3 Linear Regression

```
>lm_model = lm(cnt ~. , data = train)
>predictions_LR = predict(lm_model, test[,1:11])
```

```
>summary(lm_model)
```

```
Call:
lm(formula = cnt ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-3906.2  -358.1    65.5   441.5  2966.5 

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1446.07     278.72   5.188 2.98e-07 ***
season2       929.34     194.81   4.770 2.35e-06 ***
season3       863.24     242.23   3.564 0.000397 ***
season4      1620.77     211.36   7.668 7.85e-14 ***
yr1          2021.76      66.11  30.580  < 2e-16 ***
mnth2         118.31     161.18   0.734 0.463251    
mnth3         490.11     184.92   2.650 0.008269 ** 
mnth4         389.83     271.41   1.436 0.151480    
mnth5         682.42     298.65   2.285 0.022687 *  
mnth6         468.91     314.07   1.493 0.136003    
mnth7          42.91     350.73   0.122 0.902664    
mnth8         335.51     341.81   0.982 0.326748    
mnth9         932.88     303.33   3.075 0.002205 ** 
mnth10        451.05     280.60   1.607 0.108523    
mnth11        -85.90     267.23  -0.321 0.748004    
mnth12        -66.41     212.52  -0.312 0.754780    
holiday1     -667.62     205.11  -3.255 0.001203 ** 
weekday1      157.20     122.94   1.279 0.201563    
weekday2      199.79     119.05   1.678 0.093875 .  
weekday3      335.88     121.46   2.765 0.005876 ** 
weekday4      312.42     120.32   2.597 0.009666 ** 
weekday5      359.63     123.93   2.902 0.003857 ** 
weekday6      392.45     119.23   3.291 0.001060 ** 
workingday1       NA         NA      NA       NA    
weathersit2  -409.49      88.87  -4.608 5.06e-06 ***
weathersit3 -2010.77     233.37  -8.616  < 2e-16 ***
temp         3172.89    1482.32   2.140 0.032751 *  
atemp        1485.15    1536.55   0.967 0.334191    
hum         -1586.90     346.98  -4.573 5.92e-06 ***
windspeed   -2289.58     508.21  -4.505 8.09e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 778.1 on 555 degrees of freedom
Multiple R-squared:  0.8439,	Adjusted R-squared:  0.836 
F-statistic: 107.2 on 28 and 555 DF,  p-value: < 2.2e-16
```

The R square and adjusted R square values are around 83.6% wich means that the input variables in this model can explain 83% of the target variable.

Values with the *** are very important in predictiong the output of the model, where as ** are comparatively less significant and the values with no * are not significant in deriving the predictor variable.

2.5.4 KNN imputation moel

```
> library(class)
> KNN_Predictions = knn(tr[, 1:11], te[, 1:11], tr$cnt, k = 2)
> KNN_Predictions=as.numeric(as.character((KNN_Predictions)))
```

The KNN model finds the nearest neigbhors and tries to predict target value.

These are the various models developed on the given data. The Data is divided into train and test. The model is developed on the train data and the test model is fit into it to predict the target variables. Later the actual and predicted values of target variable are compare to get the error and accuracy.

The model with least error and best accuracy will be taken as the model for future use.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using predictive performance as the criteria to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

3.1.1 Mean Absolute Error (MAE)

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

In R, Define MAPE usinf function

```
> MAPE = function(y, yhat){
      mean(abs((y - yhat)/y)*100)
    }
```

```
> MAPE(test[,12], predictions_DT) #Mape for Decision Tree
[1] 22.54898
> MAPE(test[,12], RF_Predictions) #Mape For Random Forest at n=200
[1] 16.9911
MAPE(test[,12], predictions_LR)   #MApe for Linear Regression
[1] 18.00231
MAPE(te[,12], KNN_Predictions)    #Mape for KNN method
[1] 22.14738
```

3.1.2 Accuracy

It is the ratio of number of correct predictions to the total number of predictions made.

Accuracy= number of correct predictions / Total predicitions made

It can also be calculated from MAE as

Accuracy = 100 – MAE

1.Accuracy fro **Decision tree** = 100-22.55 = **77.45%**

2. Accuracy for **Random forest** = 100-16.99 = **83.01%**
3. Accuracy for **Linear Regression** = 100-18.00=**82%**
4. Accuracy for **KNN method** = 100-22.15 =**77.85%**

3.2 Model Selection

From the values of Error and accuracy, it is seen that all the models perform similar in terms of Error. As such any model can be used, but **Random forest** gives better results compared to other algorithms.

APPENDIX
COMPLETE R CODE

```
rm(list=ls())

#x="C:\Users\Chaitrali\Downloads\Data\data01s2l1\Edwisor-Project"
#gsub("\","/",x)

setwd("C:/Users/Chaitrali/Downloads/Data/data01s2l1/Edwisor-Project")
install.packages("Hmisc")
x=c("ggplot2", "DMwR", "corrgram", "Hmisc", "rpart", "randomForest")
lapply(x, require, character.only = TRUE)
rm(x)

#load  the data
Rental_train=read.csv("day.csv" , header= T)[,-1]
str(Rental_train)

#convert into required data type
fac= 1:8
nu = 9:15
Rental_train[,fac]= lapply (Rental_train[, fac], as.factor)
Rental_train[,nu]= lapply (Rental_train[, nu], as.numeric)

str(Rental_train)



#Missing value Analysis
missing_val = data.frame(apply(Rental_train,2,function(x){sum(is.na(x))}))
rm(missing_val)
#no Missing value present

###############################################Outlier
Analysis################################################
# ## BoxPlots - Distribution and Outlier Check
numeric_data = Rental_train[,nu]
#excluding the output quantities
cnames = colnames(numeric_data)[1:4]
cnames



for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"), data = Rental_train)+
         stat_boxplot(geom = "errorbar", width = 0.5) +
         geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                      outlier.size=1, notch=FALSE) +
         theme(legend.position="bottom")+
          labs(y=cnames[i],x="y")+
          ggtitle(paste("Box plot of rental count for",cnames[i])))
}


## Plotting plots together
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3, gn4,ncol=2)

# #Remove outliers using boxplot method
df = Rental_train
Rental_train = df


#Replace all outliers with NA and impute
#create NA on outliers
for(i in cnames){
  val = Rental_train[,i][Rental_train[,i] %in% boxplot.stats(Rental_train[,i])$out]
  print(length(val))
  print(val)
  Rental_train[,i][Rental_train[,i] %in% val] = NA
}

 Rental_train[1,11]=NA
 missing_val = data.frame(apply(Rental_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$Missing_percentage/nrow(Rental_train)) * 100
 missing_val = missing_val[order(-missing_val$Missing_percentage),]
 row.names(missing_val) = NULL
 missing_val = missing_val[,c(2,1)]

# # ###Check which method is suitable for missing value imputation
#Mean Method
 Rental_train$hum[is.na(Rental_train$hum)] = mean(Rental_train$hum, na.rm = T)
 Rental_train[1,11]
#Median Method
 Rental_train$hum[is.na(Rental_train$hum)] = median(Rental_train$hum, na.rm = T)
 Rental_train[1,11]
 Rental_train[1,11]=NA
#
# # kNN Imputation
Rental_train = knnImputation(Rental_train, k = 5)
#
# #Mean=0.6291165
# #Median=0.6283082
# #KNN=0.6268289
```

```
#since mean gives closest value, choose Mean method for imputation
#Put original value
#Rental_train[1,11]=0.805833

sum(is.na(Rental_train))
Rental_train$hum[is.na(Rental_train$hum)] = mean(Rental_train$hum, na.rm = T)
Rental_train$windspeed[is.na(Rental_train$windspeed)] = mean(Rental_train$windspeed, na.rm = T)

rm(i, gn1,gn2,gn3,gn4, val)
##########################3Feature Selection###########################

## Correlation Plot
corrgram(Rental_train[,nu], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

#Finding the correlation between the numeric variables
num_cor=round(cor(numeric_data), 3)


# ## Chi-squared Test of Independence
factor_data = Rental_train[,fac]
colnames(factor_data)

for (i in 1:8){
  for (p in 1:8){
    print(names(factor_data)[i])
    print(names(factor_data)[p])
    print(chisq.test(table(factor_data[,p],factor_data[,i]), p-value))
    }
  }

#to check the degree of association of categorical variable
library(GoodmanKruskal)
plot(GKtauDataframe( factor_data), corrColors = 'blue')

#Since cnt is the sum of casual and registered users we drop them too

reduced_train = subset(Rental_train,
                    select = -c(casual, registered, dteday))

####################################Model Development#########################
set.seed(123)
train_index = sample(1:nrow(Rental_train), 0.8 * nrow(Rental_train))
colnames(Rental_train)

train = Rental_train[train_index,-c(1,13,14)]
test = Rental_train[-train_index,-c(1,13,14)]
```

```
#write.csv(train, "Trial1.csv", row.names = F)
#write.csv(test, "Trailte1.csv", row.names = F)

MAPE = function(y, yhat){
  mean(abs((y - yhat)/y*100))
}


fit = rpart(cnt  ~. , data = train, method = "anova")
predictions_DT = predict(fit, test[,-12])
MAPE(test[,12], predictions_DT)


#Random Forest
RF_model = randomForest(cnt ~.  , train, importance = TRUE, ntree = 200)
RF_Predictions = predict(RF_model, test[,-12])
MAPE(test[,12], RF_Predictions)

importance(RF_model, type = 1)

#Linear Regression
lm_model = lm(cnt ~. , data = train)
summary(lm_model)
predictions_LR = predict(lm_model, test[,1:11])
MAPE(test[,12], predictions_LR)

##KNN Implementation
library(class)
#Predict test data
KNN_Predictions = knn(train[, 1:9], test[, 1:9], train$cnt, k = 2)
KNN_Predictions=as.numeric(as.character((KNN_Predictions)))
#Calculate MAPE
MAPE(test[,10], KNN_Predictions)
#Gaussian
poi= glm(formula = cnt~. , family = "gaussian" , train )
poi_pre = predict(poi, test[, 1:11])
MAPE(test[,12], poi_pre)
```

Reference

- ✓ Websites:
    - www.edwisor.com
    - www.analyticsvidhya.com
    - www.tuitorialspoint.com
- ✓ Books
    - R in action- Robert Kabarcoff