Bike Renting Project

Madhurima Biswas

7 February 2019

Contents

I. Introduction:

1.1. Problem Statement

Bike rental systems are a flexible transport service where users can rent a two-wheeler vehicle without going through the hassle of buying or maintaining one's own bike. We are provided daily rental data spanning two years 2011-2012. The objective of this case is to Predication of bike rental count on daily based on the environmental and seasonal settings, so that it helps in better management of the bike rental systems to organize & update their bikes for customers.

1.2. Data

The task is to build 'predictive regression' models, which will predict the bike rental count, based on the various factors given in the data during the year 2011-2012.

Given below is a sample of the data set that we are using to predict the rental count:

Table 1.1: Bike Renting sample data (year 2011-12)

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit
1	01-01-11	1	0	1	0	6	0	2
2	02-01-11	1	0	1	0	0	0	2
3	03-01-11	1	0	1	0	1	1	1
4	04-01-11	1	0	1	0	2	1	1
5	05-01-11	1	0	1	0	3	1	1
6	06-01-11	1	0	1	0	4	1	1

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
0.226957	0.22927	0.436957	0.1869	82	1518	1600
0.204348	0.233209	0.518261	0.0895652	88	1518	1606

In the table above, we have the following 15 independent variables, using which we have to predict the Bike Rental Count:

Table 1.2: Predictor Variables

No.	Independent Variables
1	instant
2	dteday
3	season
4	yr
5	mnth
6	holiday
7	weekday
8	workingday
9	weathersit
10	temp
11	atemp
12	hum
13	windspeed
14	casual
15	registered

We have in total 7 categorical variables, 8 numeric variables & one Date type variable.

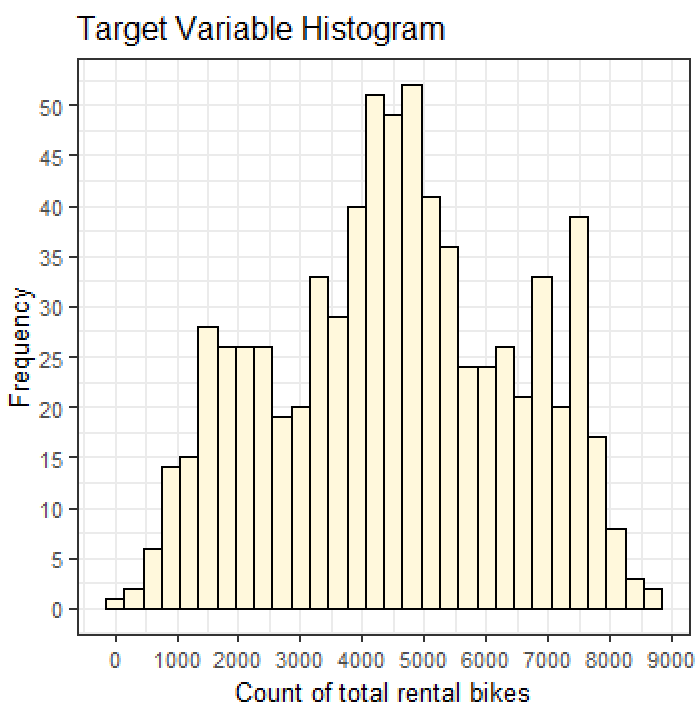
II. Methodology

2.1. Exploratory Data Analysis:

The objective first is to study each feature available in the data and try to assess some patterns and understand the dimensions & properties of the data by exploring it visually. It helps us in understanding the nature of data in terms of distribution of the individual variables/features, finding missing values, relationship with other variables and many other things.

2.1.1. Univariate Analysis:

A. Dependent Target Variable: “cnt”



Since our target variable is continuous, we can visualize it by plotting its histogram.

Observation:

- The curve of the frequency distribution of “cnt” variable seems close to normal distribution curve, having mean = 4504 & median = 4548.
- Removing outliers might help reduce the slight skewness in data.
- Range: [22, 8714]

B. Independent Numeric Variables:

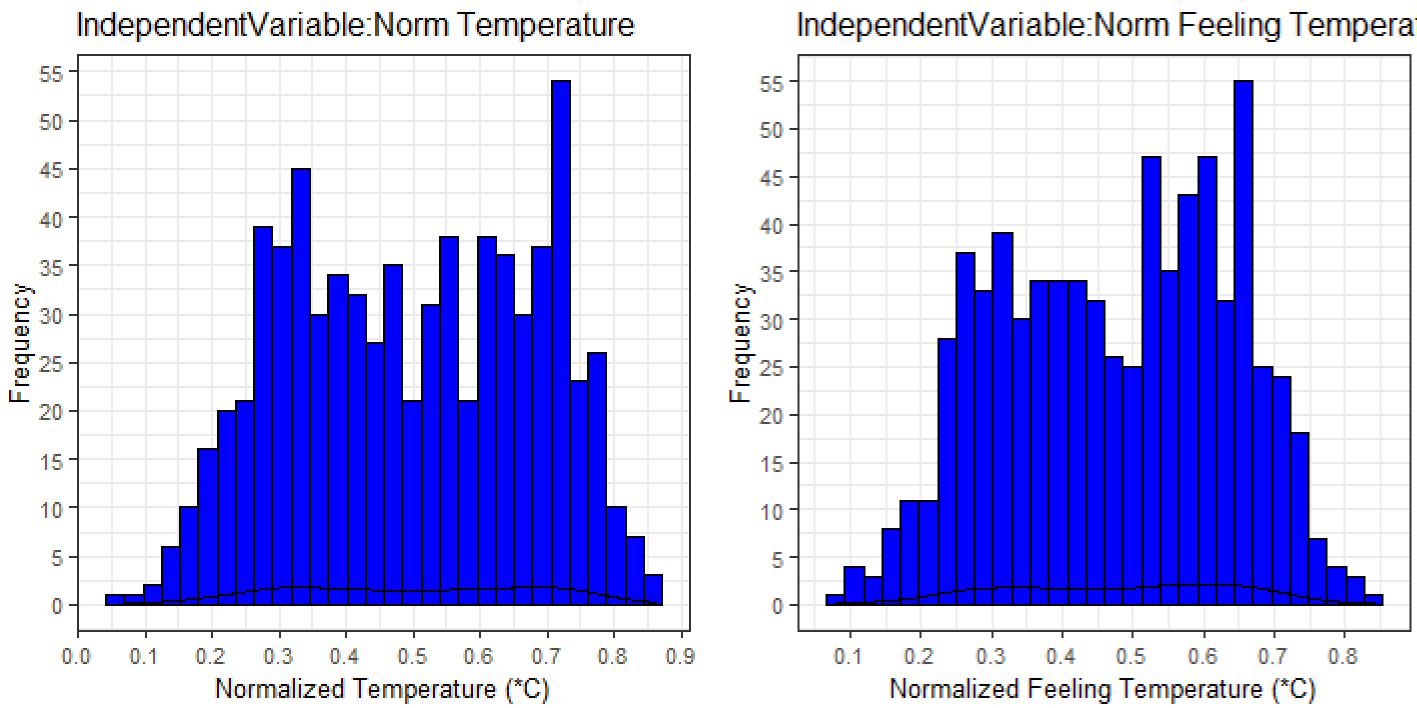


Fig.2.2

Observation:

- We see a wide range of temperature during years 2011-12; however, there is no clear-cut pattern.
- “Humidity” has a mean of 0.62 and “Windspeed” has a mean of 0.19 in 2011-12. Mostly are within a smaller range and the curve is slight skewed for both.

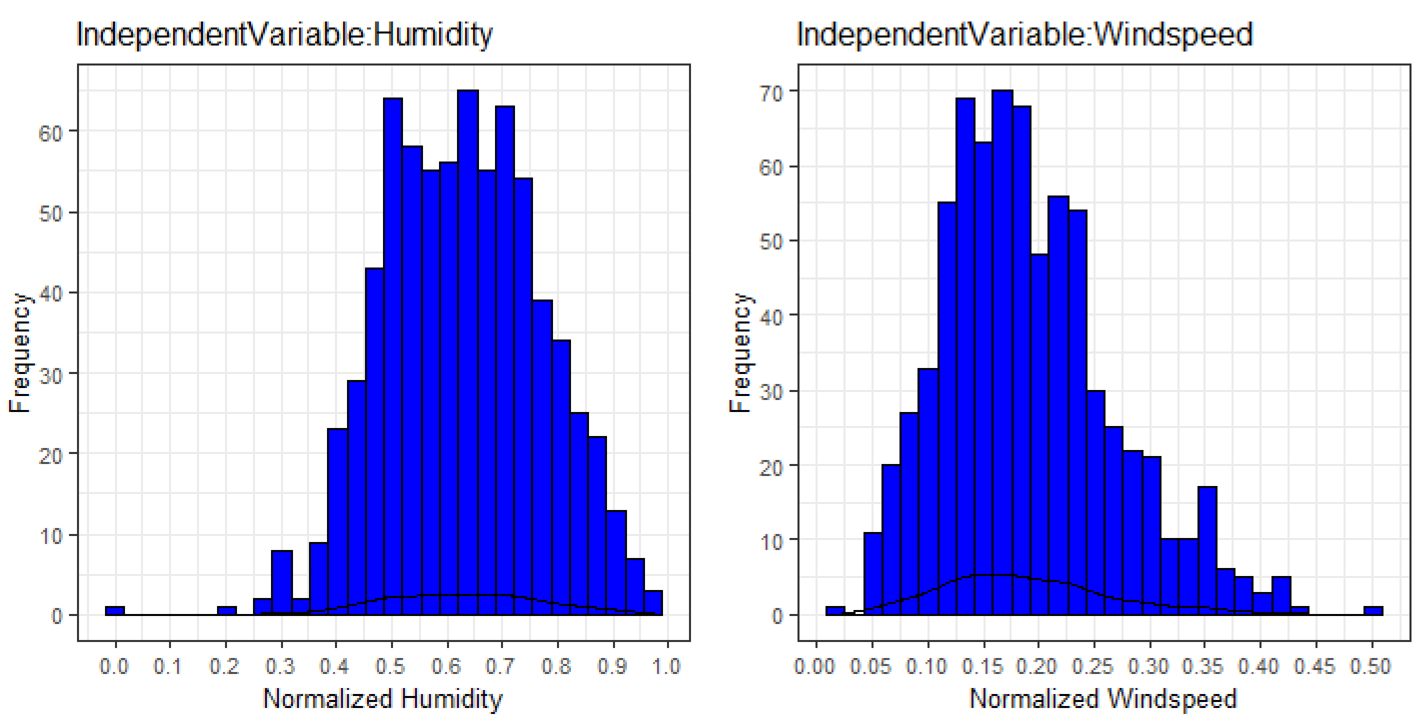
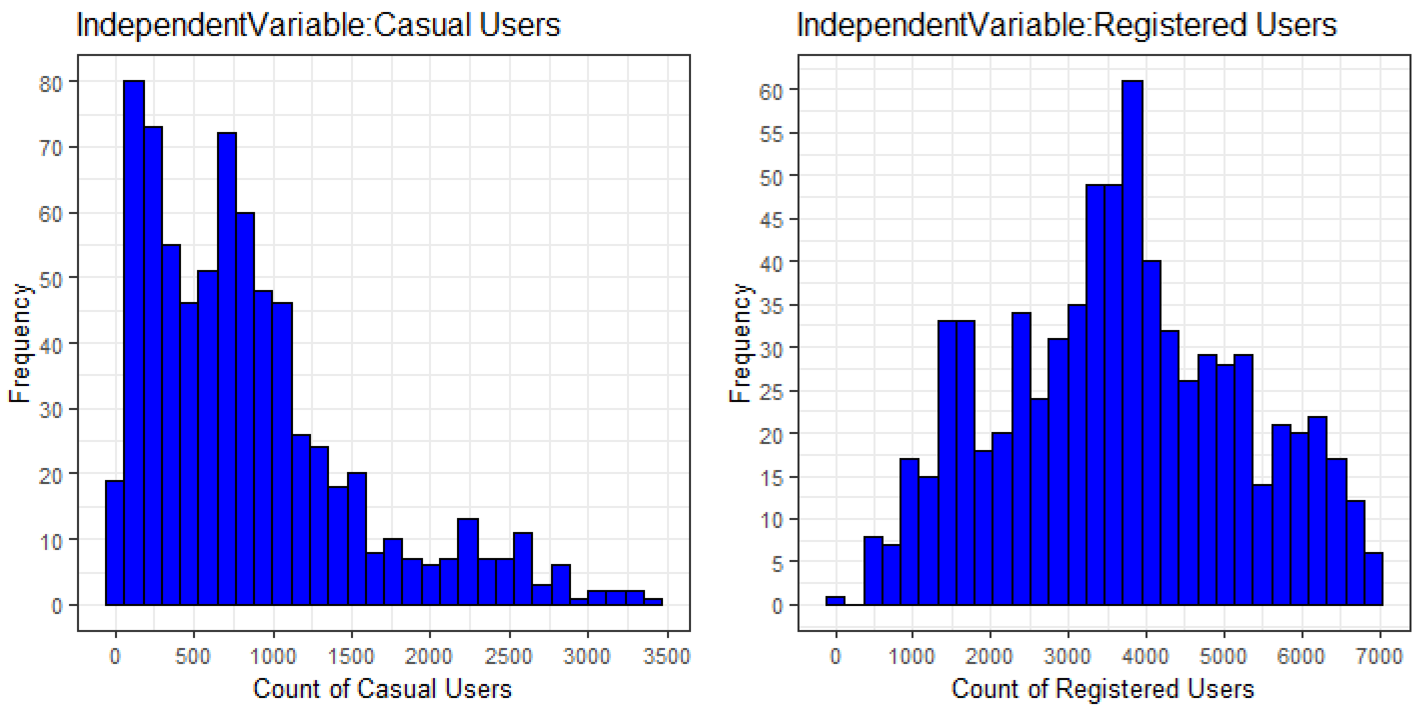


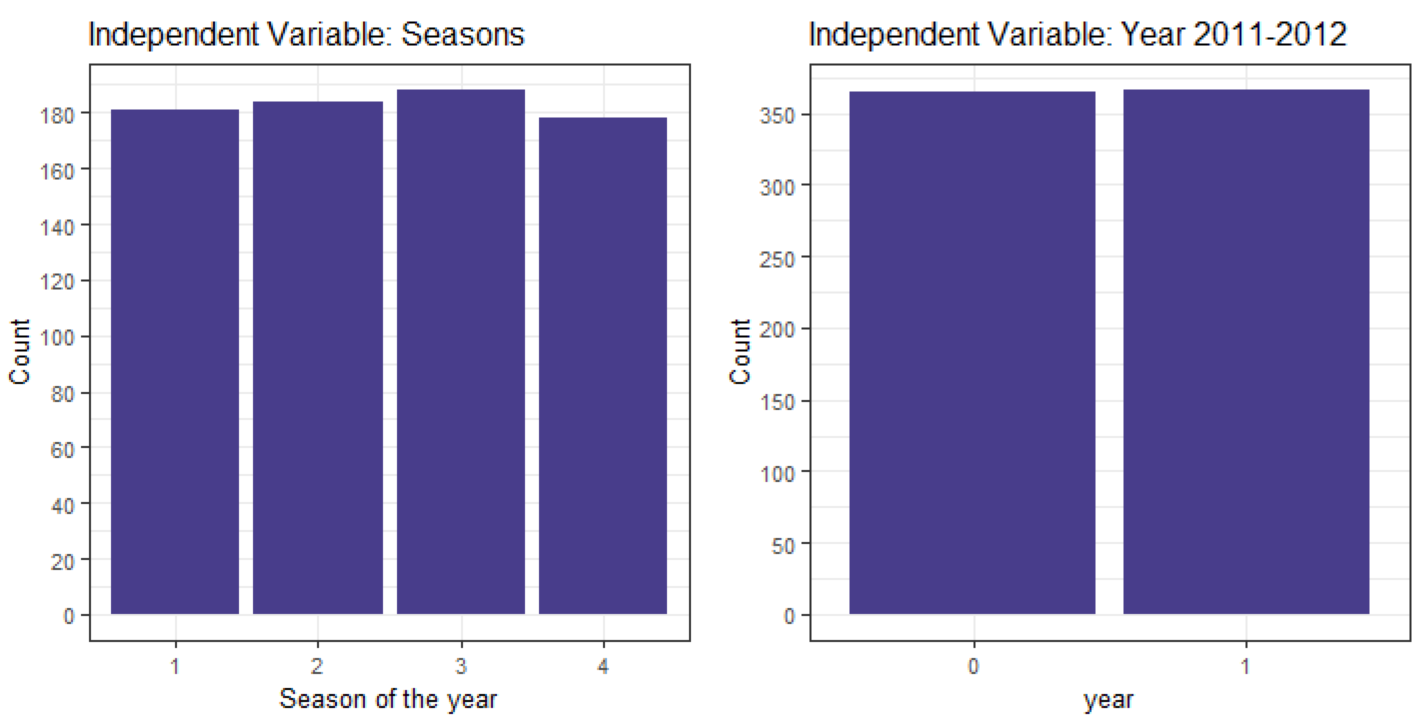
Fig.2.3



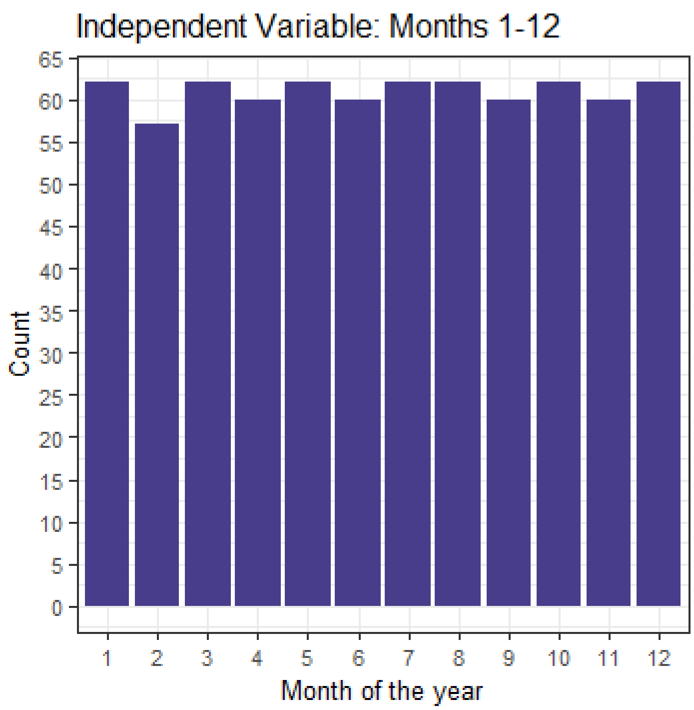
- We observe that most “Casual” per day are lesser in number and it is positively skewed frequency curve. “Registered” users per day have a mean of 3656 over 2011-12.

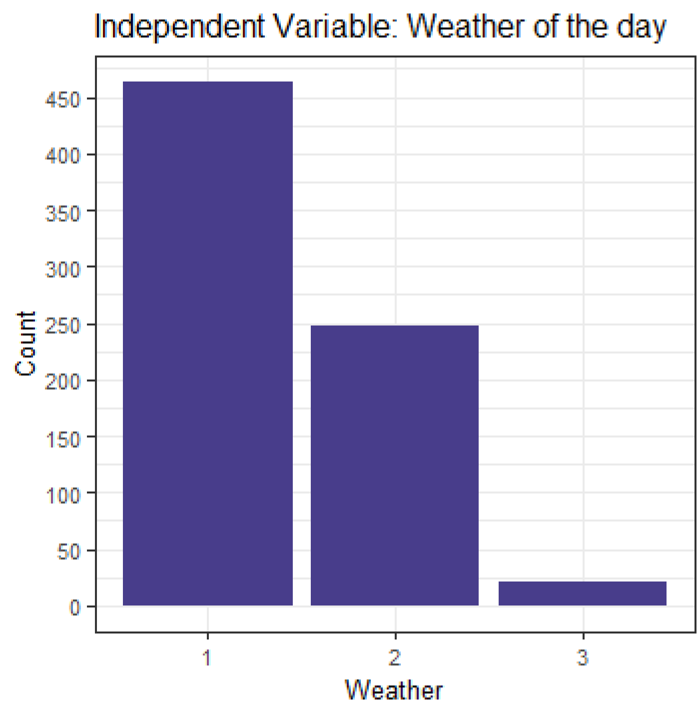
C. Independent Categorical Variables:

Bar graphs for the categorical variables in the data as follows:

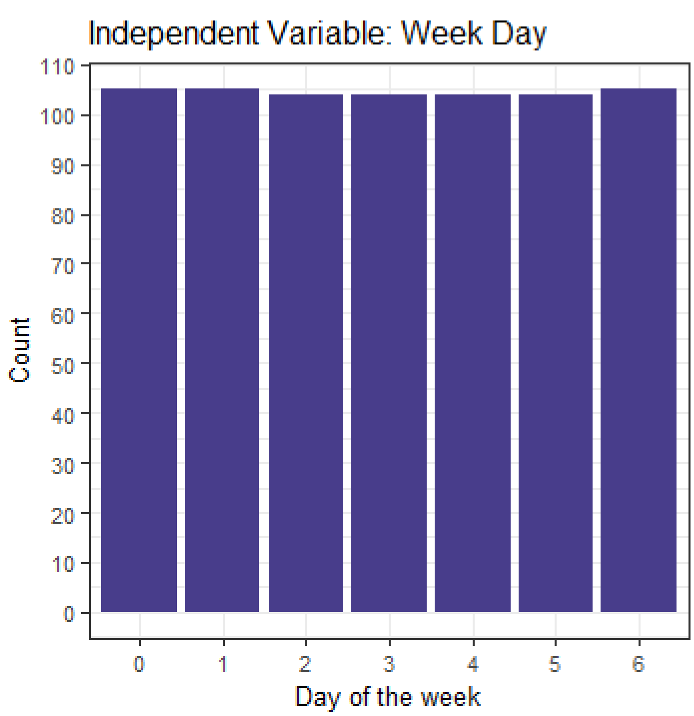


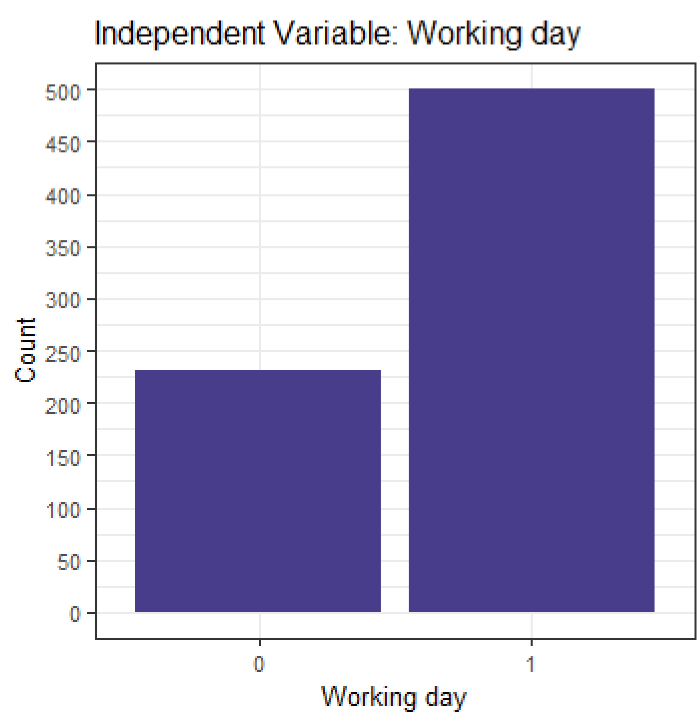
- We observe that “fall” season was compartively longer and “winter” was shortest in the years 2011-12.
- The data is well distributed within year 2011 & 2012 and we can hope to get less error due to data imbalance yearly.





- “Mnth” variable has correct data as February has least days and others have 30/31 days. The weather in 2011-12 was mostly clear or few clouds or partly cloudy.
- “week days” seem properly distributed and obviously there are more working days than holidays and weekends.

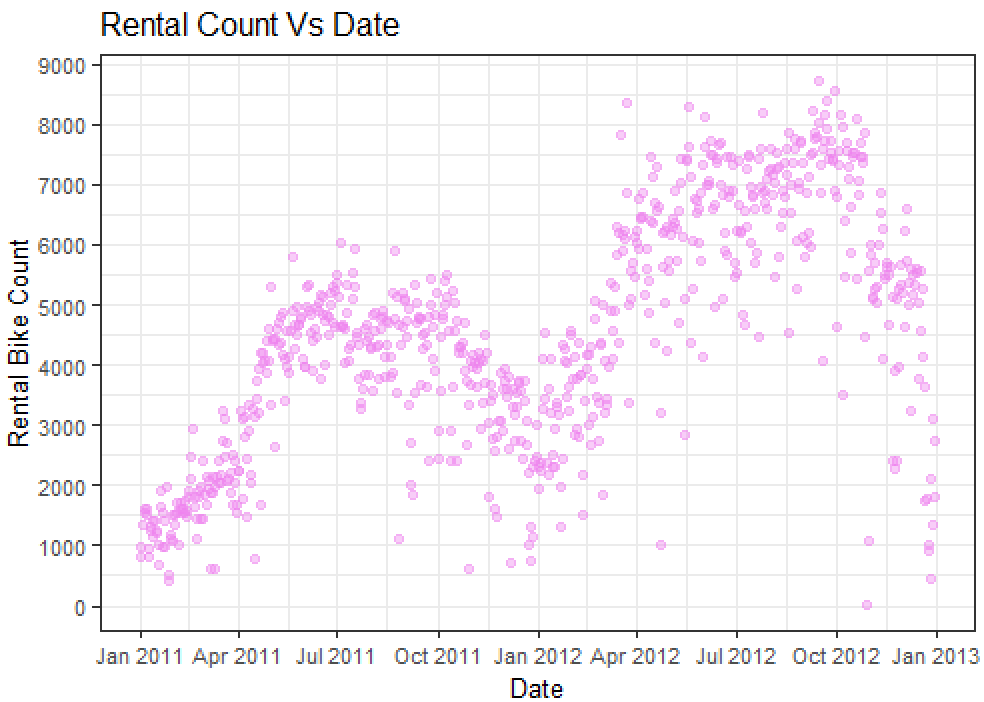




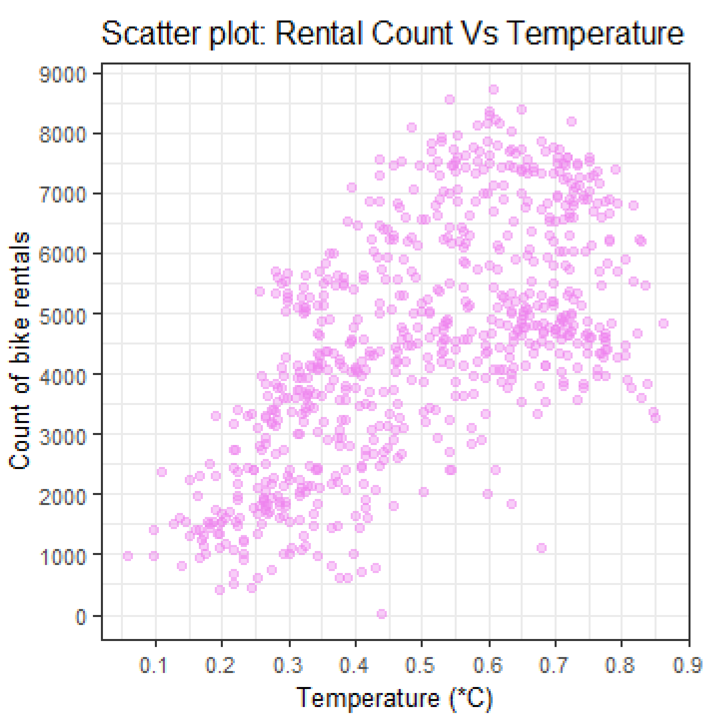
2.1.2. Bivariate Analysis:

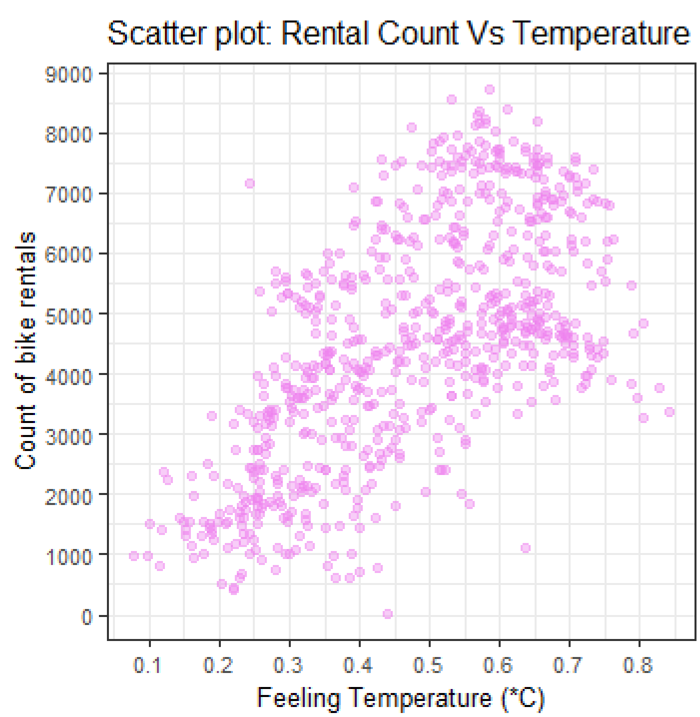
After looking features individually, let’s explore the independent variables with respect to the target variable using scatter plots to discover hidden relationships between the independent variable and the target variable and use those findings in missing data imputation and feature engineering.

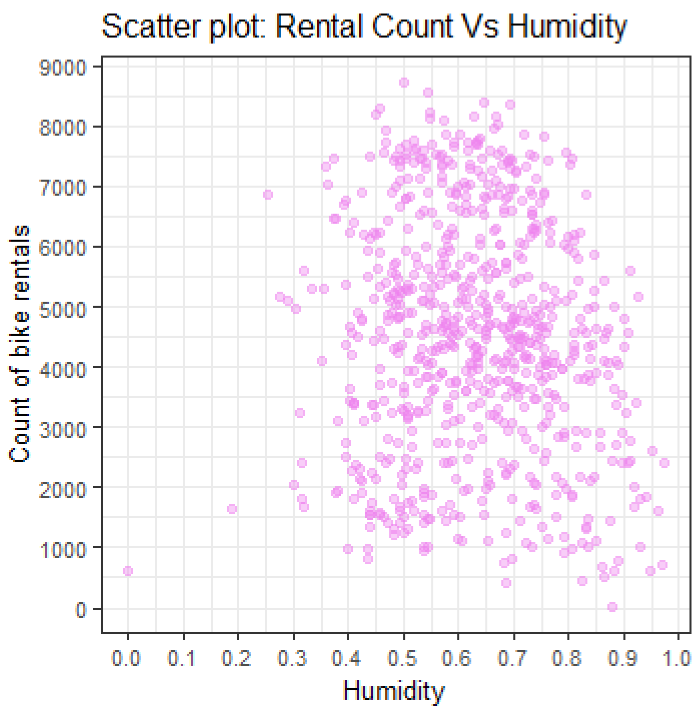
A. Dependent target Variable Vs Independent Variables:

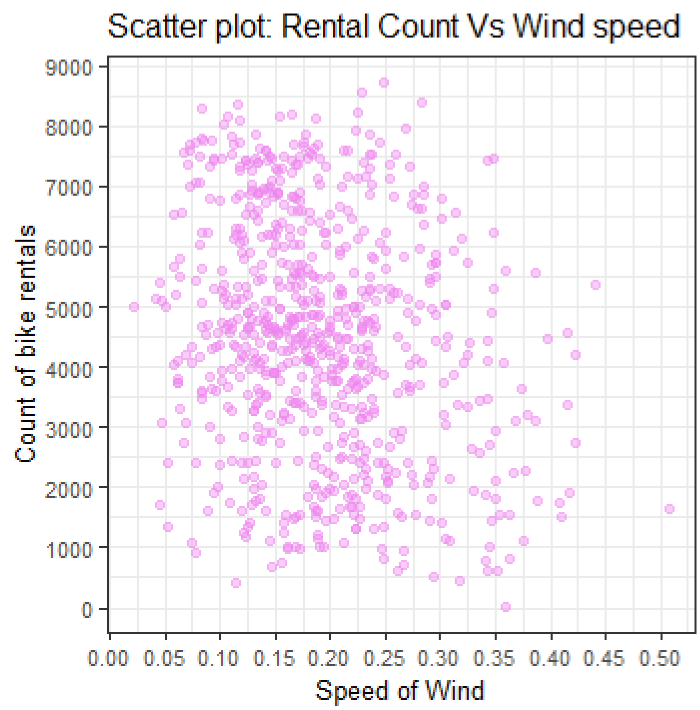


- We observe there are more bikes rented during the year 2012; it maybe because of the increase in popularity of the rental system after a successful year.
- There is no clear-cut pattern for temperature & rental count.

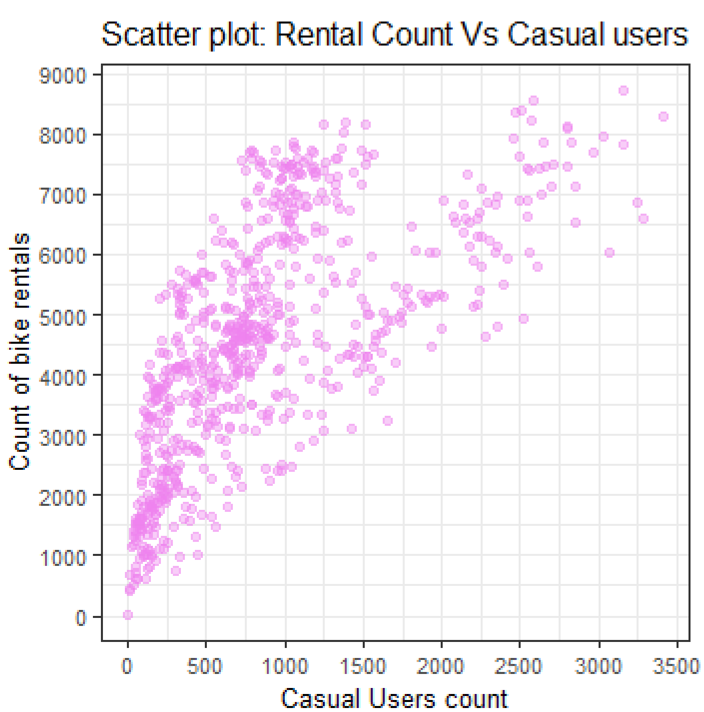


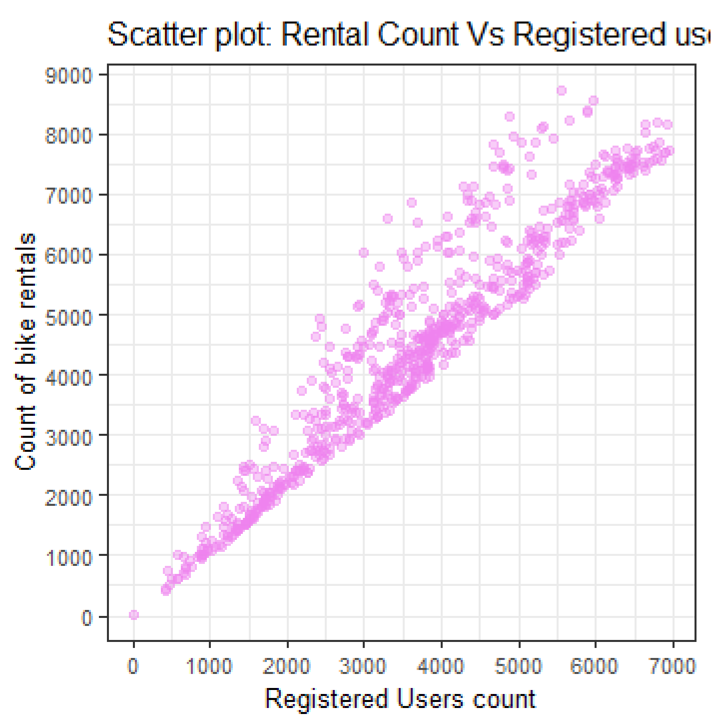


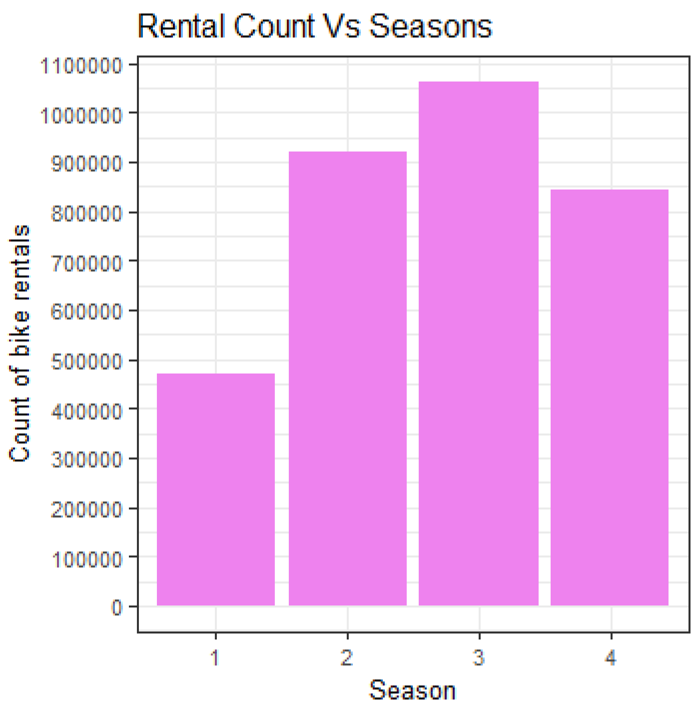


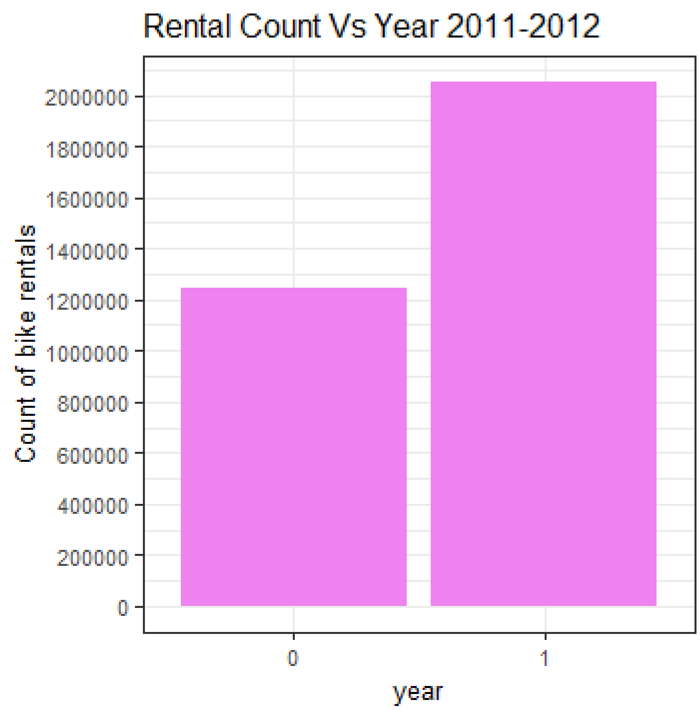


- No such clear-cut patten for humidity or Wind speed & rental count.
- We see a slight linear relationship between Casual users & rental count and a strong linear relationship between Registered users & rental count, which is kind of obvious as rental count is the sum of total users, i.e. casual as well as registered customers.

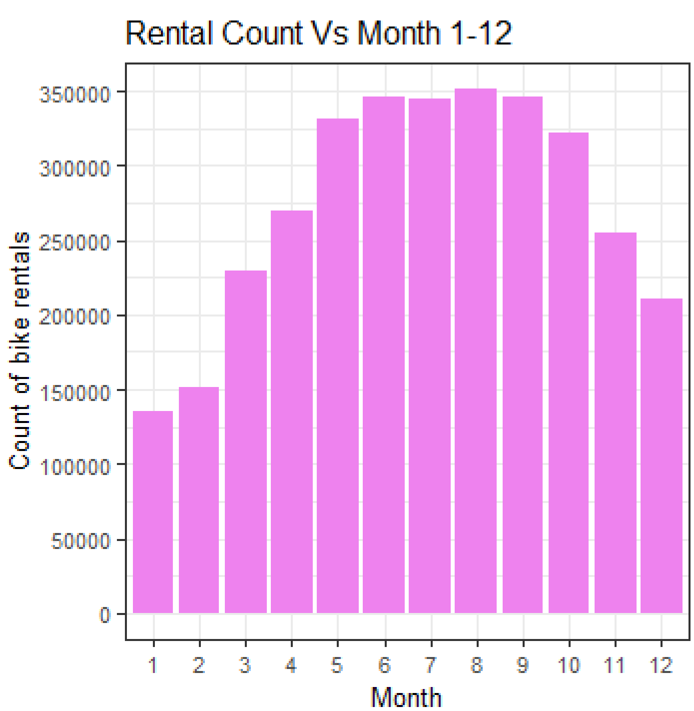


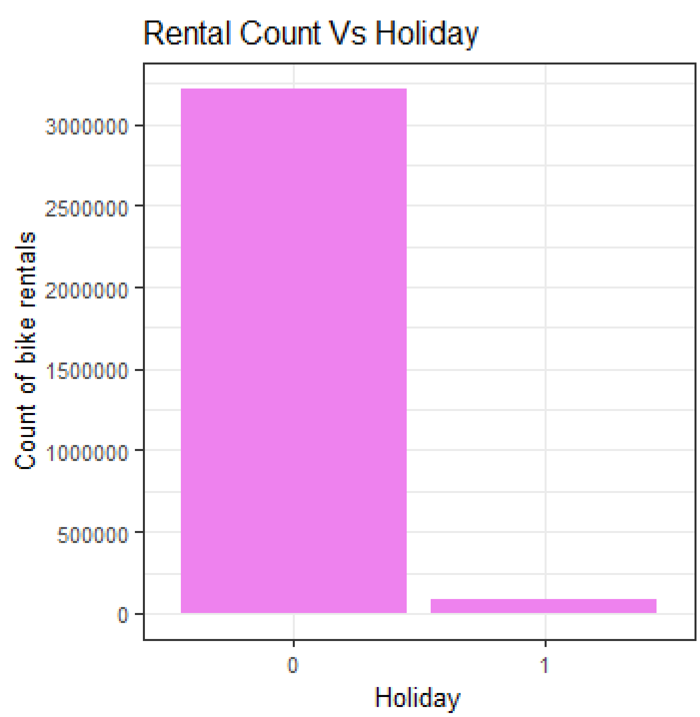


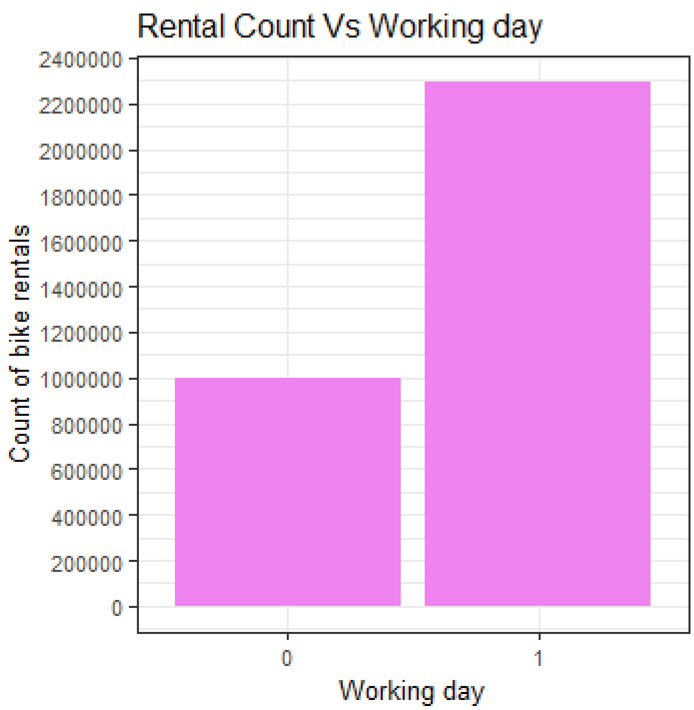


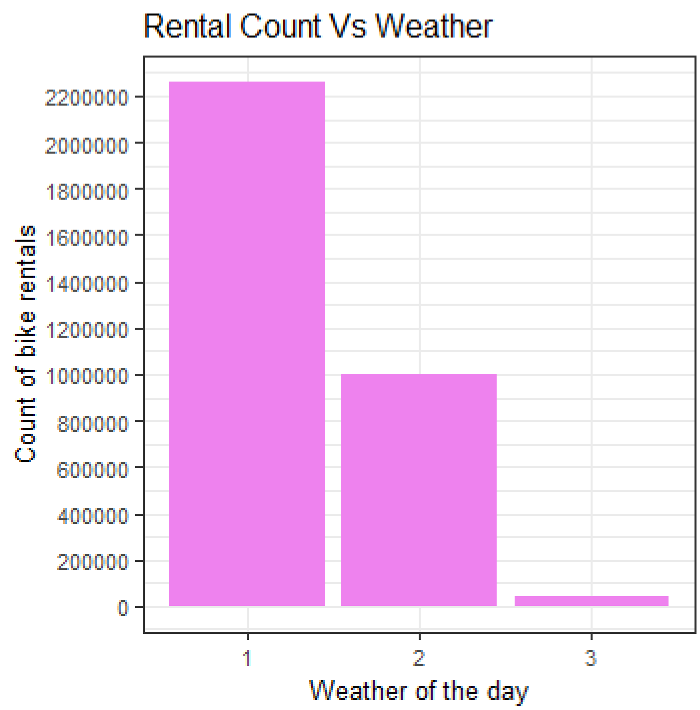


- Rental count is most in fall season and that is expected as fall season was longest in 2011-12. However, the rental count is quite low in spring season which maybe due to low temperature during the season and there maybe seasonal influences on this.
- More bikes were rented in 2012 than in 2011, which could be due to the increase in popularity of bike rental systems.
- Most bikes rented in the month of June, July, August & September.





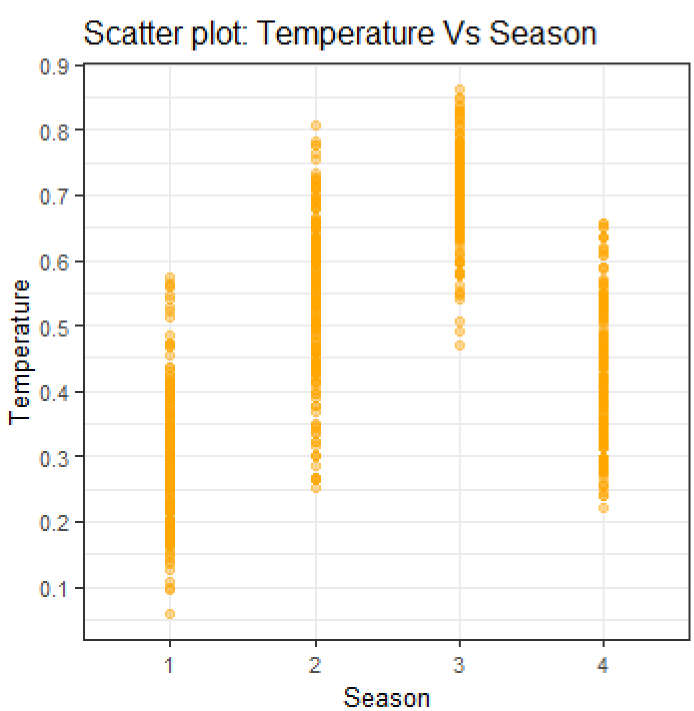


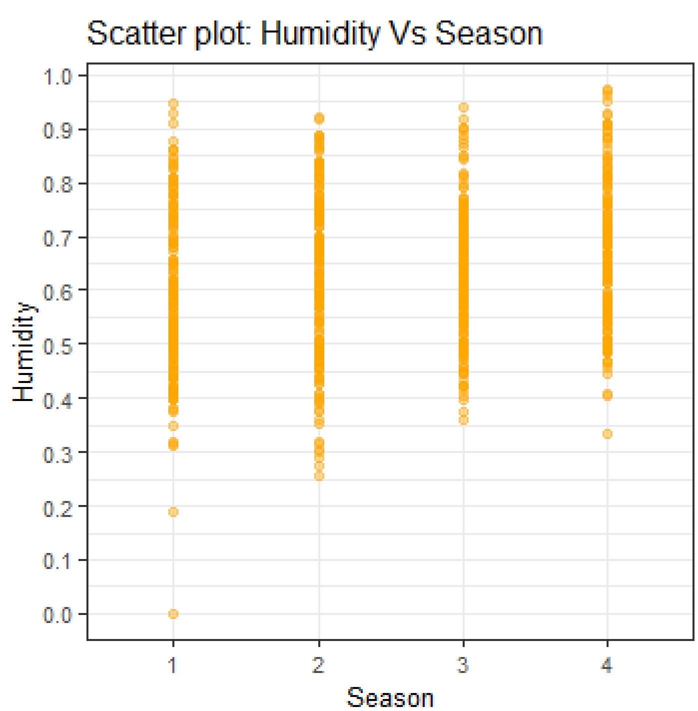


- Bikes are more rented on working days, but that is also because there are more working days in a year.
- More bikes rented on days with clear to partly cloudy weather, but that is also most days in a year.

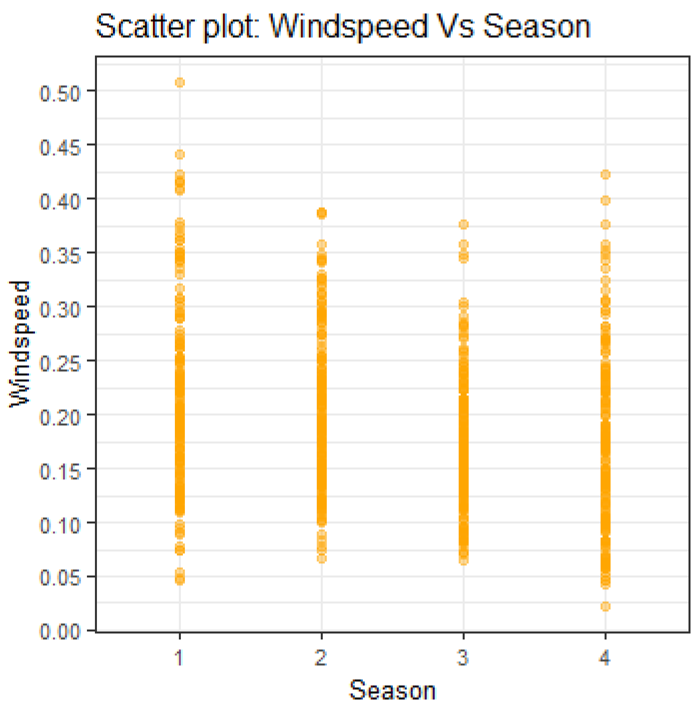
B. Independent Variable Vs Independent Variable (Interdependencies):

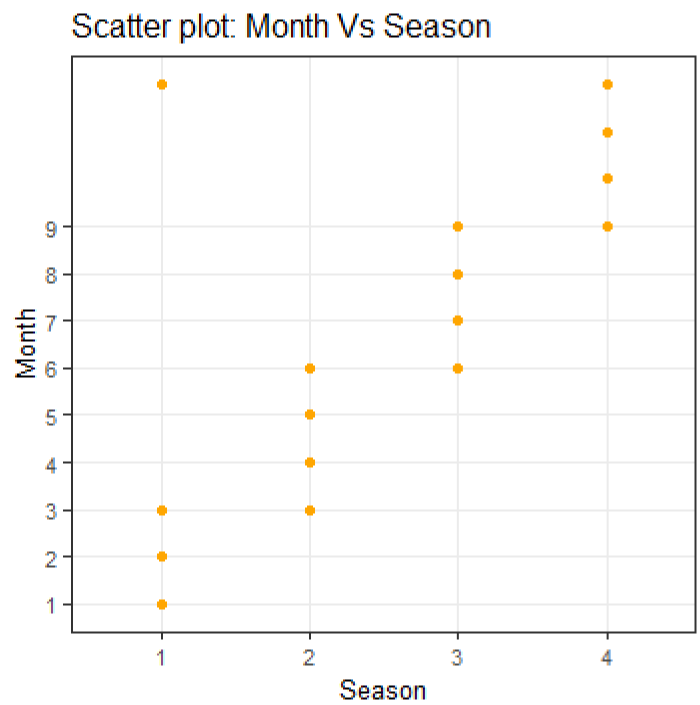
Let's explore the independent variables with respect to other independent variables using scatter plots to discover hidden relations or dependencies between them.



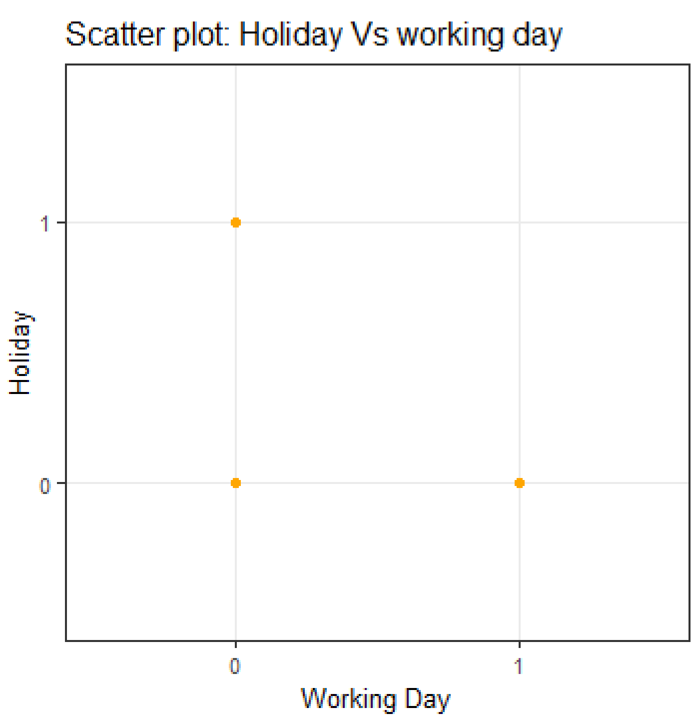


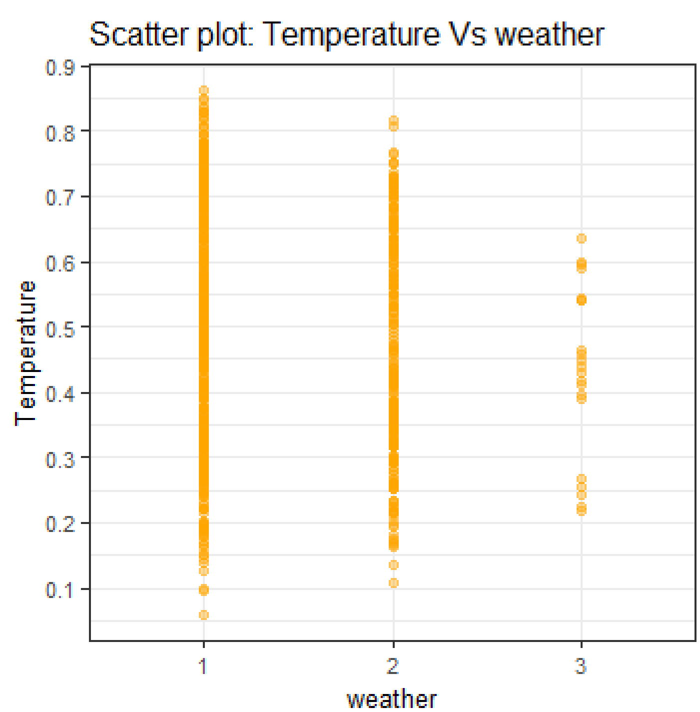
- We observe that temperature is highest in fall season and lowest in winter season.
- We don't see any dependence of humidity with seasonal changes.

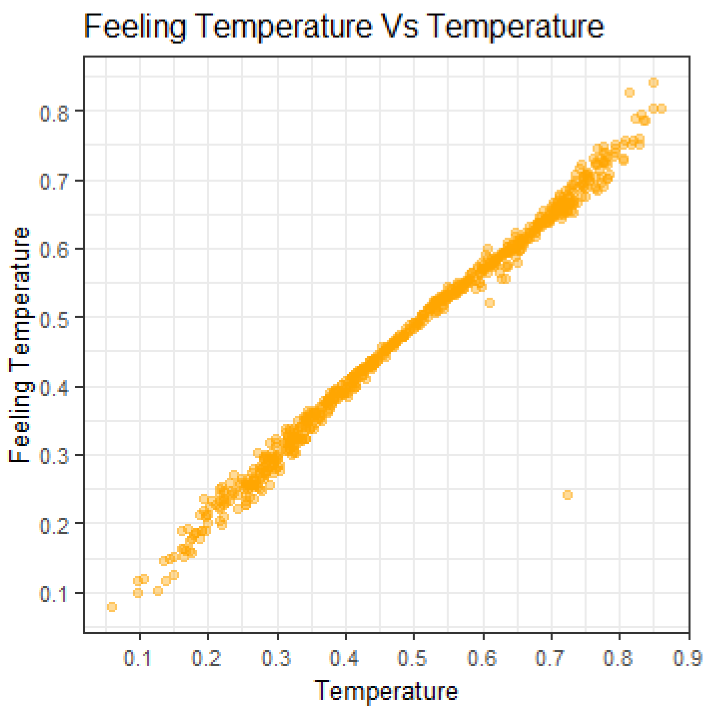


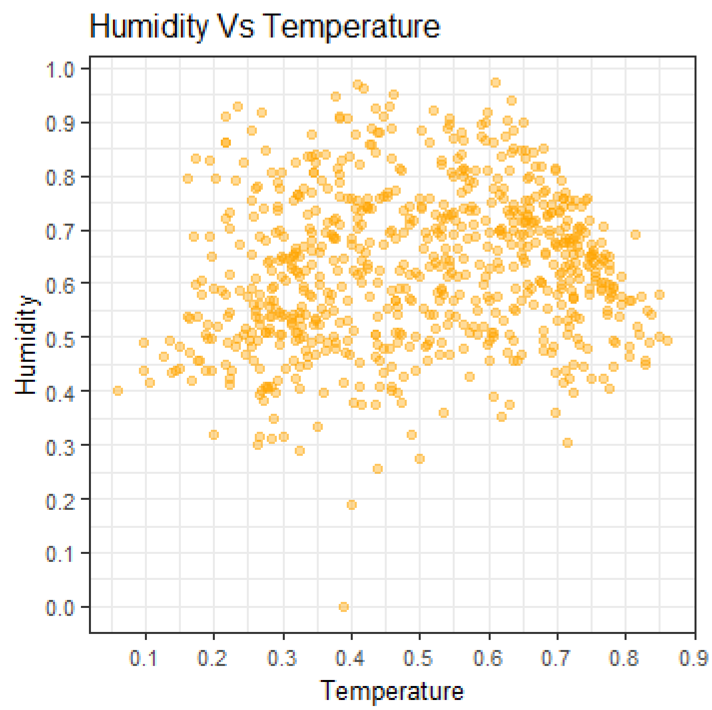


- No such clear-cut pattern of windspeed due to seasonal changes.
- December, March, June & September are the months of seasonal transition.
- There are no days in the data where it was a holiday and a working day both simultaneously, as expected.
- During clear to partly cloudy days, the temperature has a wide range.

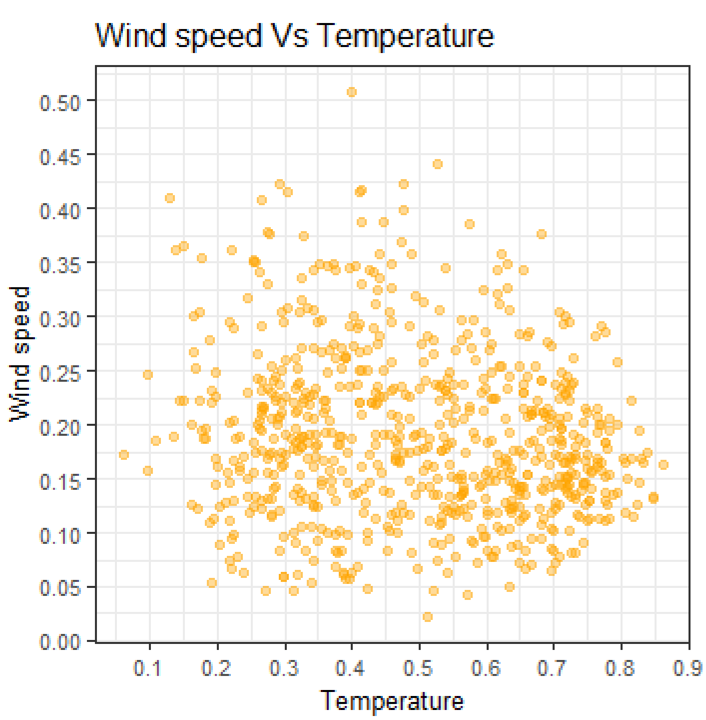


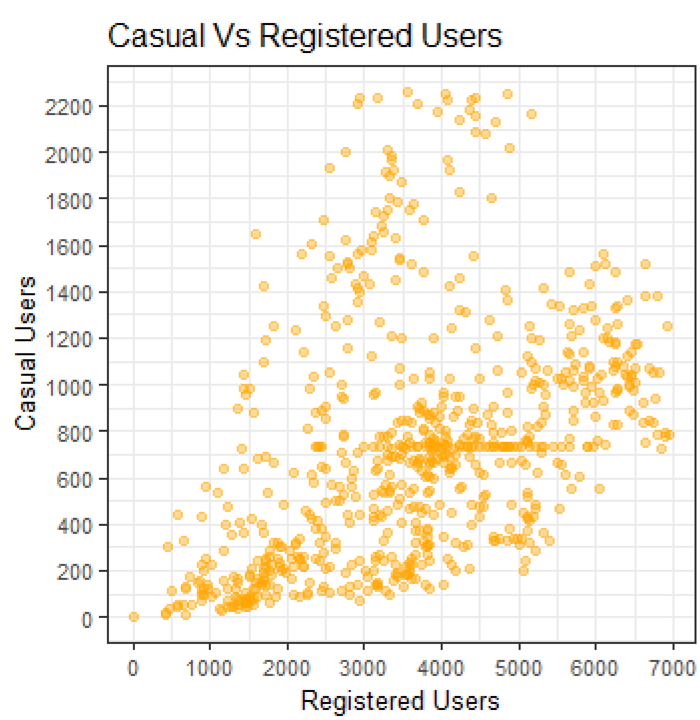


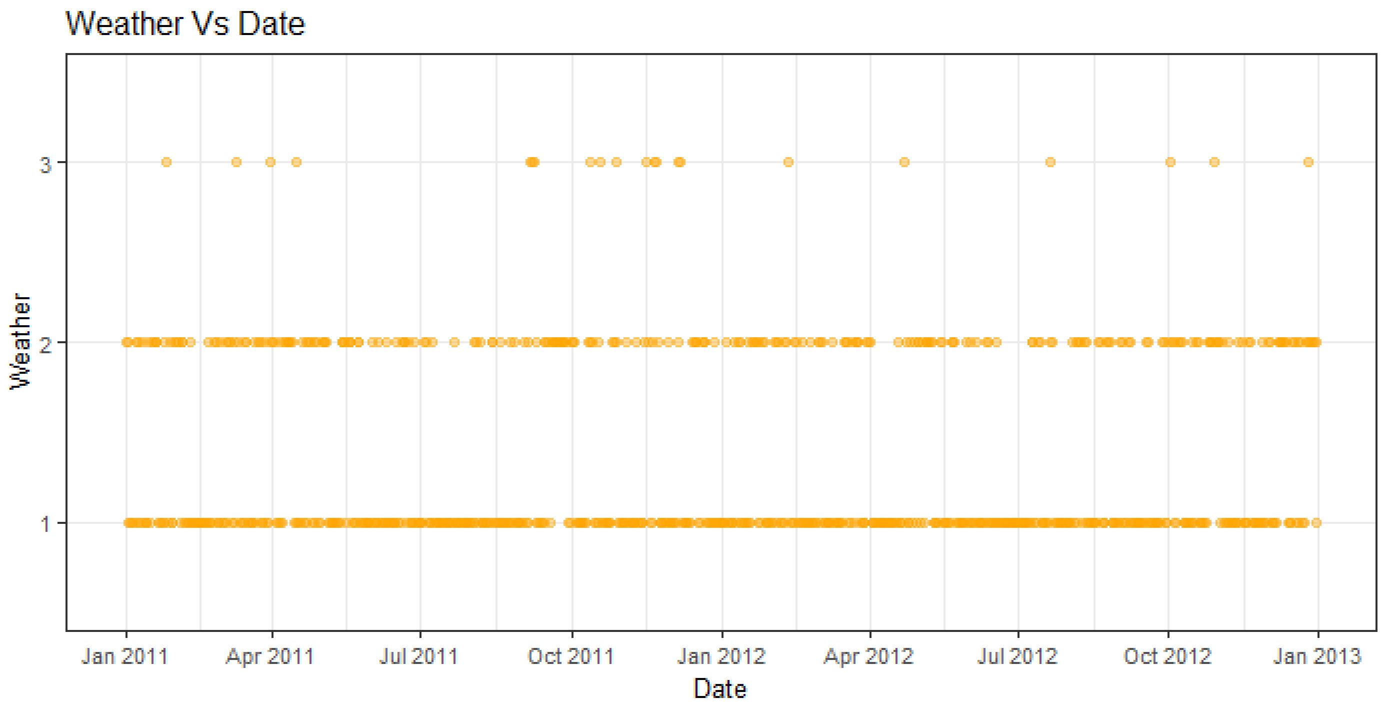




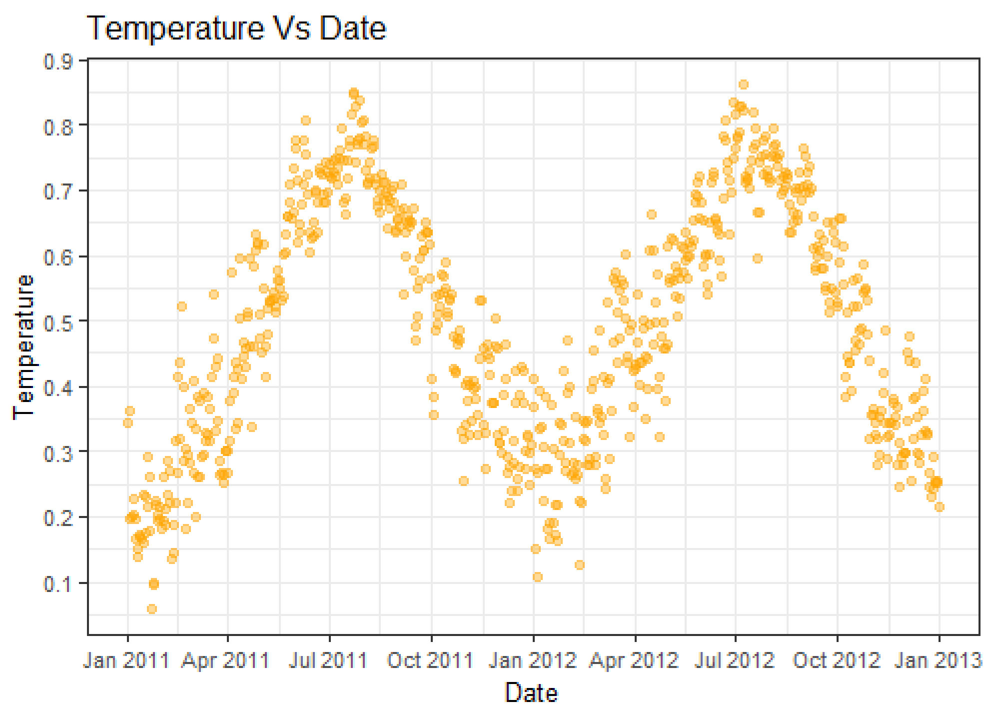
- There is a strong linear relationship between temperature & feeling temperature, as expected.
- No clear-cut pattern of humidity with temperature, except there were less dry air days in 2011-12.
- No clear-cut pattern of wind speed with temperature.
- There is a faint linear relationship between casual users & registered users, but not solid enough to be claimed.







- Most of the year, we observed weather to be clear to partly cloudy in 2011-12.



- We observe a similar pattern of temperature in year 2011 & 2012, which tells us that the temperature changes were similar in both the year in changing seasons.

2.1.3. Data Consolidation:

Since not all data are in their proper data types, we need to convert it first to proceed further.

```
#_____Data type conversion_____#
catnames = c("season","yr","mnth","holiday","weekday","workingday","weathersit")
#categorical variables
for (i in catnames) {
  data[,i] = as.factor(data[,i])
}

numnames = c("temp","atemp","hum","windspeed","casual","registered","cnt")
#numerical variables
for (i in numnames) {
  data[,i] = as.numeric(data[,i])
}

data$dteday = as.Date(data$dteday)     #It changed date "02-04-11" to "2011-04-02".
```

2.1.4. Missing Value Analysis:

Missing data can have a severe impact on building predictive models because the missing values might be contain some vital information, which could help in making better predictions. So, it becomes imperative to carry out missing data imputation.

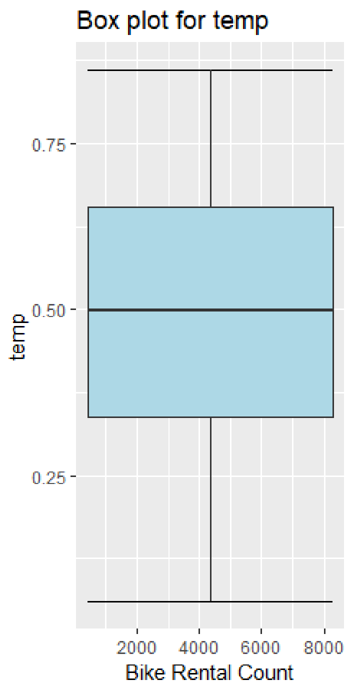
However, there are no missing values in this dataset and thus we move to next step.

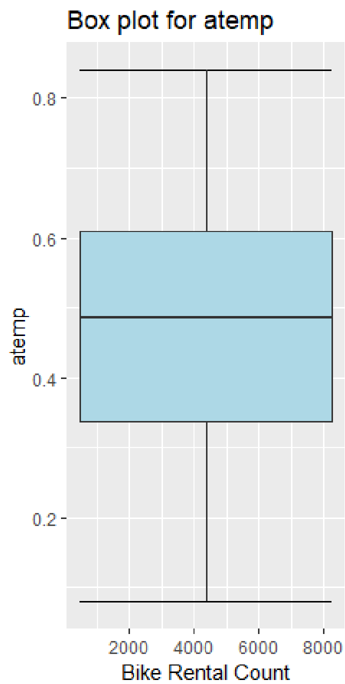
2.1.5. Outlier Analysis:

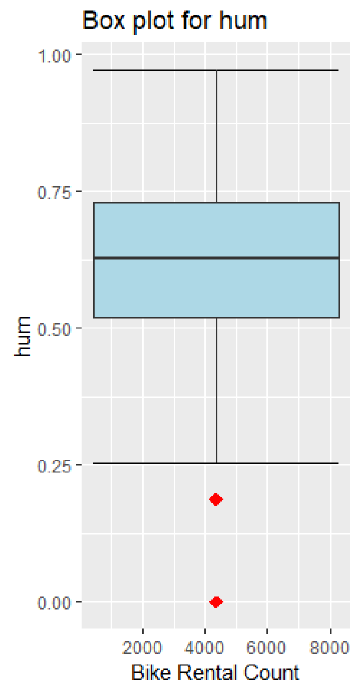
By definition, outliers are points that are distant from remaining observations. As a result, they can potentially skew or bias any analysis performed on the dataset. It is therefore important to detect and adequately deal with outliers using Box Plot method here.

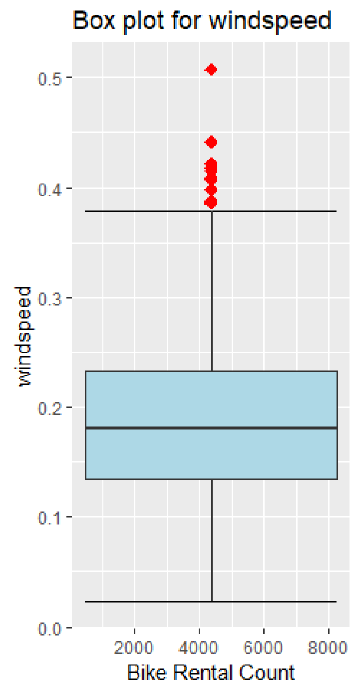
Outliers make sense only in numeric or continuous data for this dataset. The “cnt” variable consists of the labels to be used to train and test the predictive models and hence it should be left untouched by further manipulation by outlier analysis.

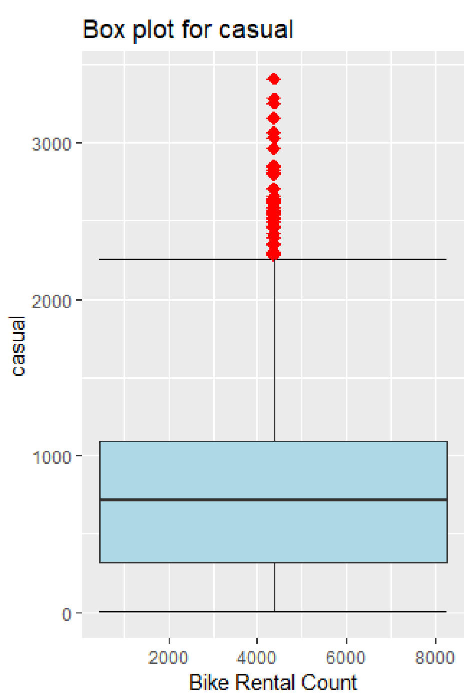
Target variable “cnt” Box plots for Independent continuous Variables (before Outlier removal):

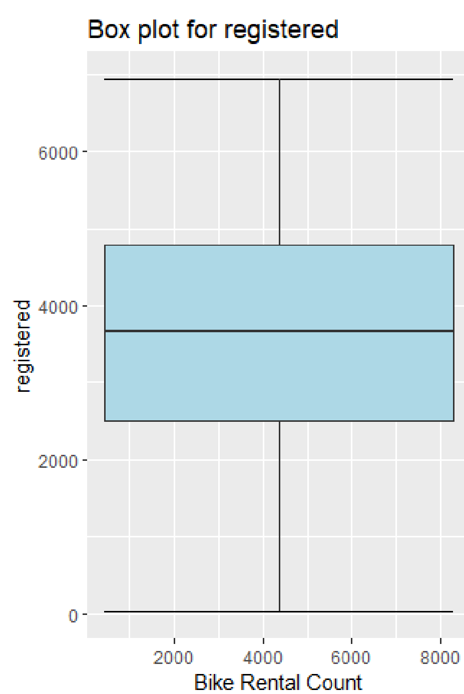


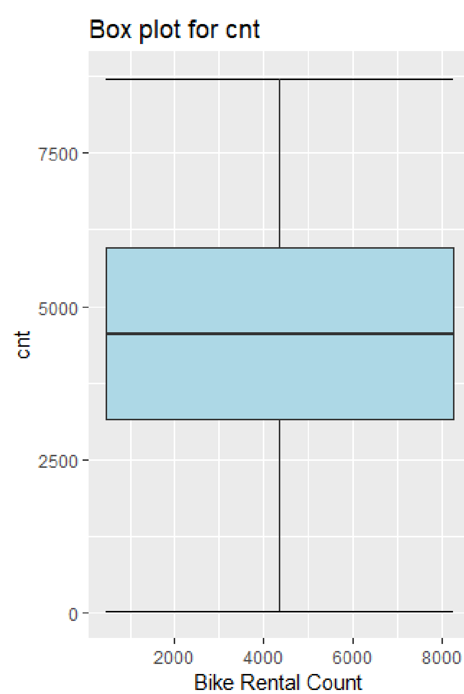




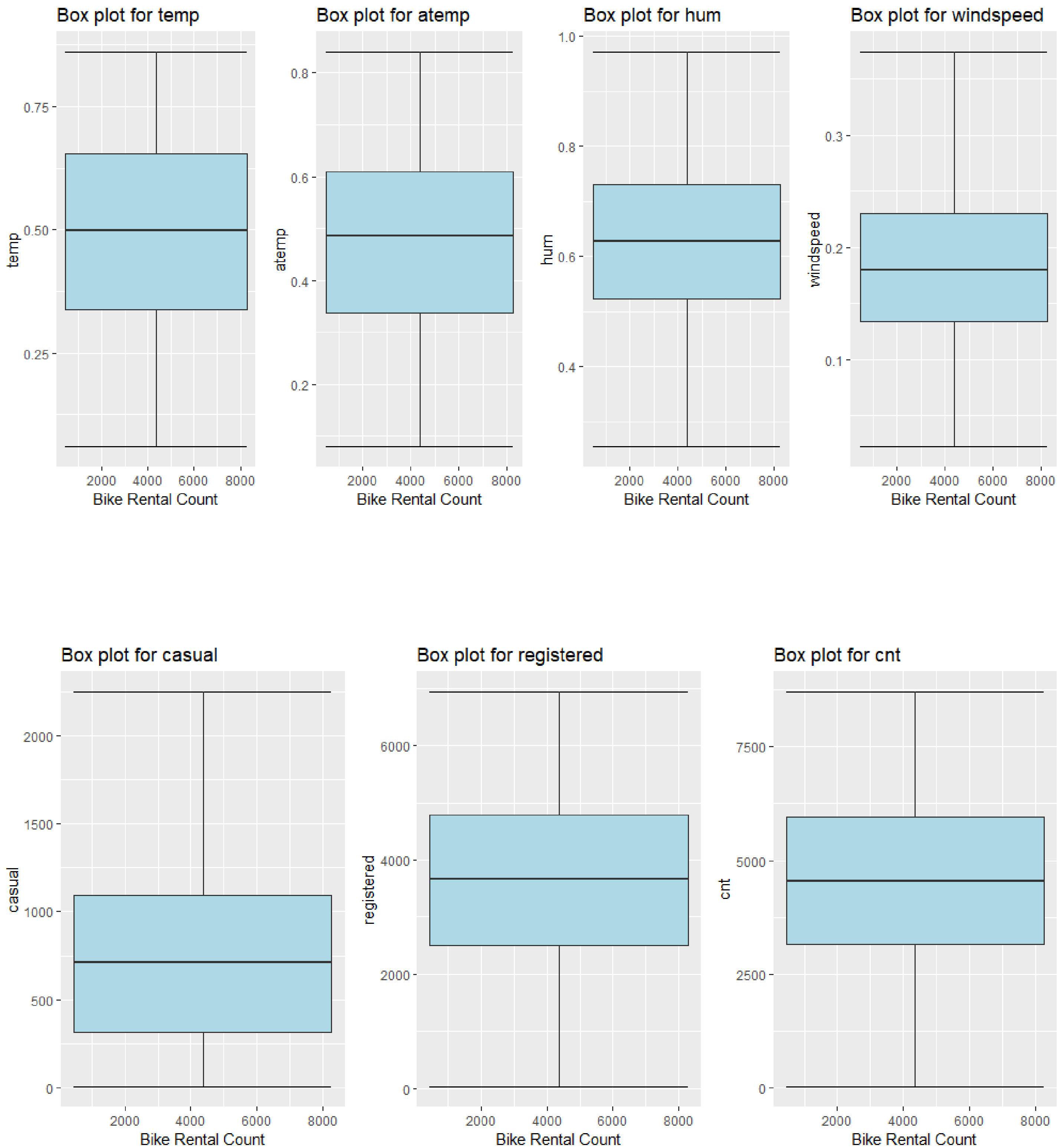








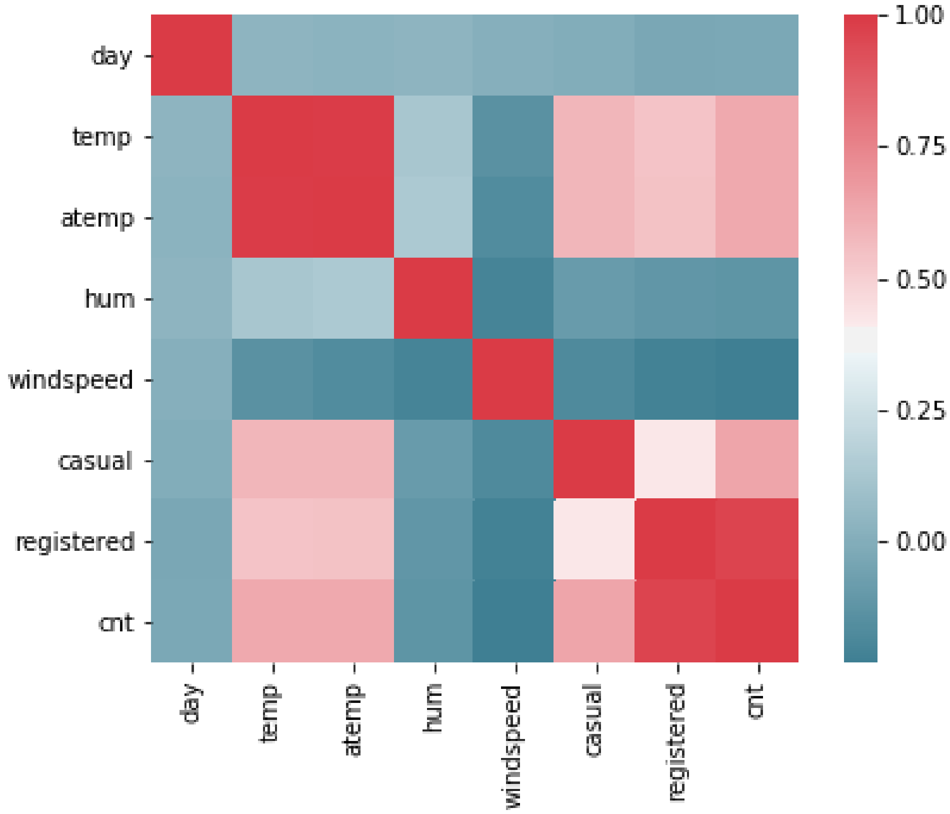
Box plots of the variables after Outlier removal:



2.1.6. Feature Selection:

It is needed that we assess the importance of each predictor variable in our analysis, as there is a possibility that many variables in our analysis are not important at all to predict the 'cnt' values.

A. Using Correlation plots:



- If |r|>0.8 for two variables, those variables are considered redundant variables and one of them can be removed from the dataset.
- Output: "temp" & "atemp" variables are highly positively correlated as expected after performing the pre-processing of the data.
- Output: "cnt" & "registered" variables are highly positively correlated as expected after performing the pre-processing of the data.

B. Using Chi-square test of Independence (relationship between categorical variables):

(Dependencies amongst Independent Categorical variables)

```
#######Chi-square Test of Independence (within Categorical Variables)
for(i in catnames){
 for(j in catnames){
   if(i!=j){
    print(names(data[i]))
    print(paste0(" Vs ", names(data[j])))
    print(chisq.test(table(data[,j],data[,i])))
   }}}
```

- If p-value<0.05 (Reject Null Hypothesis) => Target variable depends on the independent variable.
- If p-value>0.05 (Do Not Reject Null Hypothesis) =>Target variable & independent variable are independent of each other.
- Output:

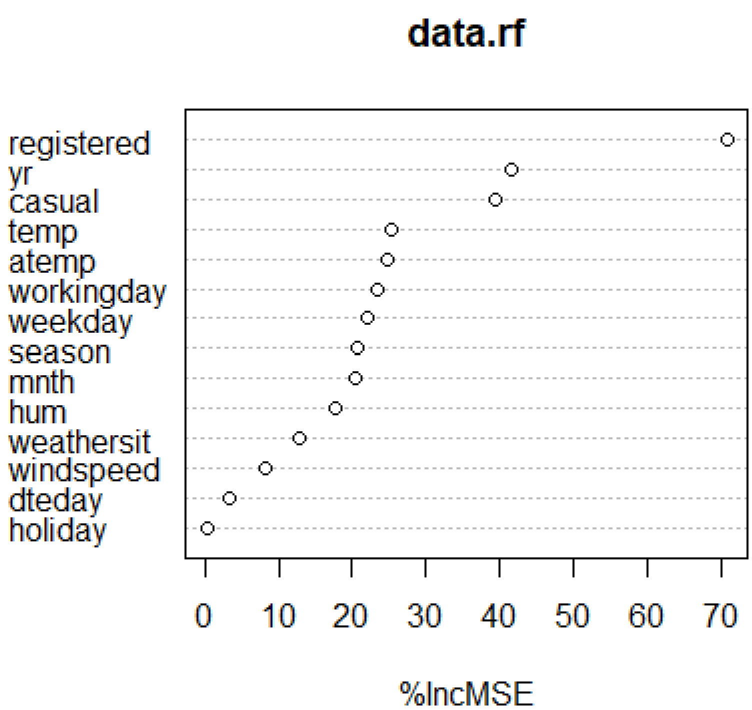
```
[1] "season"
[1] " Vs yr"

	Pearson's Chi-squared test

data:  table(data[, j], data[, i])
X-squared = 0.0041569, df = 3, p-value = 0.9999

[1] "season"
[1] " Vs mnth"

	Pearson's Chi-squared test

data:  table(data[, j], data[, i])
X-squared = 1765.1, df = 33, p-value < 2.2e-16

[1] "season"
[1] " Vs holiday"

	Pearson's Chi-squared test

data:  table(data[, j], data[, i])
X-squared = 1.4961, df = 3, p-value = 0.6832
.
.
.
```

- "workingday"-"holiday","weekday"-"workingday","weekday"-"holiday" & "mnth"-"season depend on each other significantly.

C. Using Random Forest Algorithm:

```
#######Using Random Forest Algorithm:
data.rf=randomForest(data$cnt~.,data = data, ntree=1000, keep.forest= F, importance= T)
importance(data.rf,type = 1)
```

- "holiday" variable has the least importance.



[%IncMSE is the most robust and informative measure. It is the increase in mse of predictions (estimated with out-of-bag-CV) as a result of variable j being permuted (values randomly shuffled).]

D. Using ANOVA test (comparision of Target Vs categorical variables)

anovacat = aov(cnt ~ season + yr + mnth + holiday + workingday + weekday + weathersit , data = data)

summary(anovacat)

```
Df     Sum Sq    Mean Sq  F value  Pr(>F)
season         3 950595868 316865289  436.234 < 2e-16 ***
yr             1 884008263 884008263 1217.030 < 2e-16 ***
mnth          11 187311622  17028329   23.443 < 2e-16 ***
holiday        1   3306975   3306975    4.553 0.03321 *
workingday     1   3209216   3209216    4.418 0.03591 *
weekday        5  12629845   2525969    3.478 0.00411 **
weathersit     2 185659616  92829808  127.800 < 2e-16 ***
Residuals    706 512813988    726365
```

- If p-value<0.05 (Reject Null Hypothesis) => Population means are significantly different.
- If p-value>0.05 (Do Not Reject Null Hypothesis) => Population means are not significantly different or are same.

E. Feature Selection

We should remove those features that do not contribute to predicting the target variable as it will only lead to increase in the complexity of the model and reduce interpretability of models.

1. While doing data exploration, we notice that “instant” variable is just a serial number column, so we can remove it.
2. From Chi-square test, we notice that "working day", "holiday" & "weekday" depend on each other and intuitively there is a logical connection within them.

We make a new variable using this connection between the three varibles
[Denote: 1-->weekend, 2--> working day, 3--> holiday]

```
data$day = NA
for (i in 1:nrow(data)){
 if ((data[i,7]=="0") && (data[i,5]=="0")){data[i,16] = 1}                    #weekend
 else if ((data[i,7]=="1") && (data[i,5]=="0")){data[i,16] = 2}               #working day
 else if ((data[i,7]=="0") && (data[i,5]=="1")){data[i,16] = 3}               #holiday
 else data[i,16] =NA }
```

3. "Season" has multicollinearity problem as well and it is related to "mnth", so we can remove it.
4. “casual” & “registered” are basically the target variables as their addition results to “cnt”. So, we can remove both & predict for just “cnt” variable.
5. “temp” & “atemp” are highly correlated and “atemp” variable’s importance was found out to be more. Intuitively also, feeling temperature matters more for customers who will be travelling by bikes and hence “temp” variable is redundant.

```
data= subset(data, select= -c(season,workingday,temp,casual,registered))
```

After dimensional reduction, we have 731 observationa x 10 variables in our data set.

2.1.7. Feature Scaling:

The dataset contains features that are highly varying in magnitudes, units and range. Feature Scaling (Normalization/Standardization) is a step of Data Pre-Processing, which is applied to independent variables or features of data. It helps to normalize the data within a particular range and sometimes helps in speeding up the calculations in distance-based algorithms.

However, the continuous variables in the data set was already normalized.

2.1.8. Data Sampling:

The whole dataset is divided into train and test split sets so that there is data from which the model can learn and there is a part of the data set using which we can do unbiased evaluation of the trained model.

Random sampling without replacement is used to split 80% of the data into training set and remaining 20% into test set.

```
sample.index = sample(nrow(data), 0.8*nrow(data), replace = F)   #80% data -->Train set, 20%-
-> Test set
train = data[sample.index,]
test = data[-sample.index,]
```

2.2. Modeling

2.2.1. Model Development:

The dataset of year 2011-2012 indicates that this is a supervised learning problem as there is the task of inferring a function or values from the labeled training data. Secondly, the dependent variable “cnt” is of real valued discrete type and therefore our prediction is of a quantity & it is a regression problem. Since we have many input variables, we shall perform a **multivariate regression analysis** on the given dataset.

2.2.2. Decision Tree Algorithm

Decision Trees

[Decision trees can handle both categorical and numerical variables at the same time as features. Every split in a decision tree is based on a feature. If the feature is categorical, the split is done with the elements belonging to a particular class. If the feature is contiuous, the split is done with the elements higher than a threshold. At every split, the decision tree will take the best variable at that moment. This will be done according to an impurity measure with the splitted branches. And the fact that the variable used to do split is categorical or continuous is irrelevant (in fact, decision trees categorize contiuous variables by creating binary regions with the threshold).]

```
dt=rpart(cnt~.,data = train,method= "anova")
> summary(dt)
Call:
rpart(formula = cnt ~ ., data = train, method = "anova")
  n= 584

          CP nsplit rel error    xerror       xstd
1 0.37445616      0 1.0000000 1.0051616 0.04580505
2 0.22311915      1 0.6255438 0.6603832 0.03348656
3 0.09060873      2 0.4024247 0.4239814 0.03179904
4 0.02962425      3 0.3118160 0.3290237 0.02734505
5 0.02934392      4 0.2821917 0.3120647 0.02819117
6 0.02895436      5 0.2528478 0.3120647 0.02819117
7 0.01189898      6 0.2238934 0.2660670 0.02168208
8 0.01131214      7 0.2119945 0.2668795 0.02194647
9 0.01000000      8 0.2006823 0.2633306 0.02187781

Variable importance
     atemp       mnth         yr        hum  windspeed weathersit    weekday
        34         27         25          8          4          1          1

Node number 1: 584 observations,    complexity param=0.3744562
  mean=4565.748, MSE=3745566
  left son=2 (234 obs) right son=3 (350 obs)
  Primary splits:
      atemp      < 0.4308565  to the left,  improve=0.37445620, (0 missing)
      yr         splits as  LR, improve=0.35623910, (0 missing)
      mnth       splits as  LLLRRRRRRRLL, improve=0.30009300, (0 missing)
      weathersit splits as  RLL, improve=0.07434951, (0 missing)
      hum        < 0.824394   to the right, improve=0.06695468, (0 missing)
  Surrogate splits:
      mnth      splits as  LLLRRRRRRRLL, agree=0.894, adj=0.735, (0 split)
      hum       < 0.5464585  to the left,  agree=0.625, adj=0.064, (0 split)
      windspeed < 0.06282915 to the left,  agree=0.616, adj=0.043, (0 split)
      dteday    < 29.5       to the right, agree=0.601, adj=0.004, (0 split)

Node number 2: 234 observations,    complexity param=0.09060873
  mean=3117.359, MSE=2302852
  left son=4 (126 obs) right son=5 (108 obs)
  Primary splits:
```

```
      yr         splits as  LR, improve=0.36780560, (0 missing)
      atemp      < 0.2607295  to the left,  improve=0.23258030, (0 missing)
      mnth       splits as  LLLRL--R-RRR, improve=0.19311160, (0 missing)
      hum        < 0.678777   to the right, improve=0.06662897, (0 missing)
      weathersit splits as  RLL, improve=0.06151398, (0 missing)
  Surrogate splits:
      hum       < 0.5725     to the right, agree=0.577, adj=0.083, (0 split)
      atemp     < 0.332973   to the left,  agree=0.573, adj=0.074, (0 split)
      windspeed < 0.1871895  to the right, agree=0.568, adj=0.065, (0 split)
      mnth      splits as  LRLRL--R-LRL, agree=0.564, adj=0.056, (0 split)
      weekday   splits as  LLLLLRL, agree=0.543, adj=0.009, (0 split)

Node number 3: 350 observations,    complexity param=0.2231192
  mean=5534.1, MSE=2369868
  left son=6 (164 obs) right son=7 (186 obs)
  Primary splits:
      yr         splits as  LR, improve=0.58840310, (0 missing)
      hum        < 0.834375   to the right, improve=0.15010660, (0 missing)
      weathersit splits as  RRL, improve=0.09686697, (0 missing)
      atemp      < 0.5018855  to the left,  improve=0.06263038, (0 missing)
      mnth       splits as  -LRLRRRRRRLR, improve=0.05588727, (0 missing)
  Surrogate splits:
      hum       < 0.6947915  to the right, agree=0.580, adj=0.104, (0 split)
      mnth      splits as  -RRLRLRRRRLR, agree=0.569, adj=0.079, (0 split)
      atemp     < 0.5296815  to the left,  agree=0.549, adj=0.037, (0 split)
      weekday   splits as  RLLRRRR, agree=0.546, adj=0.030, (0 split)
      windspeed < 0.1741335  to the right, agree=0.543, adj=0.024, (0 split)

Node number 4: 126 observations,    complexity param=0.02962425
  mean=2265.302, MSE=1057926
  left son=8 (75 obs) right son=9 (51 obs)
  Primary splits:
      mnth       splits as  LLLLR----RRR, improve=0.48612910, (0 missing)
      atemp      < 0.251738   to the left,  improve=0.30669750, (0 missing)
      windspeed  < 0.112571   to the right, improve=0.24712020, (0 missing)
      hum        < 0.86       to the right, improve=0.11724950, (0 missing)
      weathersit splits as  RLL, improve=0.07345125, (0 missing)
  Surrogate splits:
      windspeed < 0.120031   to the right, agree=0.746, adj=0.373, (0 split)
      atemp     < 0.298832   to the left,  agree=0.714, adj=0.294, (0 split)
      hum       < 0.611667   to the left,  agree=0.611, adj=0.039, (0 split)
      dteday    < 22.5       to the left,  agree=0.603, adj=0.020, (0 split)
      day       splits as  LLR, agree=0.603, adj=0.020, (0 split)

Node number 5: 108 observations,    complexity param=0.02895436
  mean=4111.426, MSE=1920095
  left son=10 (31 obs) right son=11 (77 obs)
  Primary splits:
      atemp      < 0.279985   to the left,  improve=0.30542030, (0 missing)
      mnth       splits as  LLLR---R-LRL, improve=0.28345620, (0 missing)
      hum        < 0.697292   to the right, improve=0.16823620, (0 missing)
      weathersit splits as  RLL, improve=0.09756212, (0 missing)
      weekday    splits as  LLLRRRL, improve=0.07717721, (0 missing)
  Surrogate splits:
      hum        < 0.4647915  to the left,  agree=0.741, adj=0.097, (0 split)
      windspeed  < 0.349942   to the right, agree=0.731, adj=0.065, (0 split)
      mnth       splits as  RRRR---L-RRR, agree=0.722, adj=0.032, (0 split)
      weathersit splits as  RRL, agree=0.722, adj=0.032, (0 split)

Node number 6: 164 observations,    complexity param=0.01131214
  mean=4276.524, MSE=648554.7
  left son=12 (29 obs) right son=13 (135 obs)
  Primary splits:
      mnth       splits as  -LLLRRRRRRLL, improve=0.23264010, (0 missing)
```

```
      hum        < 0.849375   to the right, improve=0.23168870, (0 missing)
      weathersit splits as  RLL, improve=0.18122010, (0 missing)
      atemp      < 0.5805125  to the left,  improve=0.17080540, (0 missing)
      windspeed  < 0.1265645  to the right, improve=0.07228776, (0 missing)
  Surrogate splits:
      atemp     < 0.456723   to the left,  agree=0.872, adj=0.276, (0 split)
      windspeed < 0.299444   to the right, agree=0.854, adj=0.172, (0 split)
      hum       < 0.908125   to the right, agree=0.829, adj=0.034, (0 split)

Node number 7: 186 observations,    complexity param=0.02934392
  mean=6642.93, MSE=1263643
  left son=14 (9 obs) right son=15 (177 obs)
  Primary splits:
      hum        < 0.8322915  to the right, improve=0.27309330, (0 missing)
      weathersit splits as  RLL, improve=0.13018900, (0 missing)
      atemp      < 0.4927355  to the left,  improve=0.12328470, (0 missing)
      mnth       splits as  -LLLRRRRRR-L, improve=0.07749548, (0 missing)
      windspeed  < 0.287627   to the right, improve=0.06415826, (0 missing)
  Surrogate splits:
      weathersit splits as  RRL, agree=0.968, adj=0.333, (0 split)
      windspeed  < 0.3526145  to the right, agree=0.957, adj=0.111, (0 split)

Node number 8: 75 observations
  mean=1673.933, MSE=304991.8

Node number 9: 51 observations
  mean=3134.961, MSE=894587.3

Node number 10: 31 observations
  mean=2904.516, MSE=1394240

Node number 11: 77 observations,    complexity param=0.01189898
  mean=4597.325, MSE=1309269
  left son=22 (18 obs) right son=23 (59 obs)
  Primary splits:
      hum        < 0.700625   to the right, improve=0.25817860, (0 missing)
      mnth       splits as  LLLR-----LRL, improve=0.23626120, (0 missing)
      weathersit splits as  RL-, improve=0.15559330, (0 missing)
      atemp      < 0.3134065  to the left,  improve=0.08333220, (0 missing)
      dteday     < 19.5       to the right, improve=0.07422147, (0 missing)
  Surrogate splits:
      weathersit splits as  RL-,          agree=0.792, adj=0.111, (0 split)
      mnth       splits as  RRRR-----LRR, agree=0.779, adj=0.056, (0 split)

Node number 12: 29 observations
  mean=3438.448, MSE=473523.1

Node number 13: 135 observations
  mean=4456.556, MSE=502863

Node number 14: 9 observations
  mean=4037.778, MSE=2317994

Node number 15: 177 observations
  mean=6775.395, MSE=847392.4

Node number 22: 18 observations
  mean=3544.722, MSE=1303907

Node number 23: 59 observations
  mean=4918.458, MSE=869753.4
```
We predict for test set:

```
predict.dt=predict(dt,test[,-10])
```

2.2.3. Random Forest Algorithm

[Random 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.]

```
rf = randomForest(cnt~., train, importance = TRUE, ntree = 500)
> summary(rf)
                Length Class  Mode
call              5    -none- call
type              1    -none- character
predicted       584    -none- numeric
mse             500    -none- numeric
rsq             500    -none- numeric
oob.times       584    -none- numeric
importance       18    -none- numeric
importanceSD      9    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y               584    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
```

We predict for test set:

```
predict.rf <- data.frame(predict(rf, subset(test, select = -c(cnt))))
```

2.2.4. Multiple Linear Regression

Multicollinearity is when independent variables in a regression model are correlated. It tries to inflate or resist the variance of different strong regressors in the data. Therefore, we need to do a collinearity check before performing linear regression.

```
#creating dummy variables for categorical data
factor_new = dummy.data.frame(factor_data, sep = ".")   #731 x 27
>
> #sampling#
> df = cbind(factor_new, num_data)
> #for (i in 1:ncol(df)) {
> #  df[,i] = as.numeric(df[,i])
> #}
> str(df)           # 731 X 32
'data.frame':  731 obs. of  32 variables:
 $ yr.0       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ yr.1       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.1     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ mnth.2     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.3     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.4     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.5     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.6     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.7     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.8     : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
 $ mnth.9      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.10     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.11     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth.12     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ day.1       : int  1 1 0 0 0 0 0 1 1 0 ...
 $ day.2       : int  0 0 1 1 1 1 1 0 0 1 ...
 $ day.3       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ weekday.0   : int  0 1 0 0 0 0 0 0 1 0 ...
 $ weekday.1   : int  0 0 1 0 0 0 0 0 0 1 ...
 $ weekday.2   : int  0 0 0 1 0 0 0 0 0 0 ...
 $ weekday.3   : int  0 0 0 0 1 0 0 0 0 0 ...
 $ weekday.4   : int  0 0 0 0 0 1 0 0 0 0 ...
 $ weekday.5   : int  0 0 0 0 0 0 1 0 0 0 ...
 $ weekday.6   : int  1 0 0 0 0 0 0 1 0 0 ...
 $ weathersit.1: int  0 0 1 1 1 1 0 0 1 1 ...
 $ weathersit.2: int  1 1 0 0 0 0 1 1 0 0 ...
 $ weathersit.3: int  0 0 0 0 0 0 0 0 0 0 ...
 $ dteday      : num  1 2 3 4 5 6 7 8 9 10 ...
 $ atemp       : num  0.364 0.354 0.189 0.212 0.229 ...
 $ hum         : num  0.806 0.696 0.437 0.59 0.437 ...
 $ windspeed   : num  0.16 0.249 0.248 0.16 0.187 ...
 $ cnt         : num  985 801 1349 1562 1600 ...
>
> set.seed(123)
> train_index = sample(1:nrow(df), 0.8*nrow(df))
> train.df = df[train_index,]         #584 x 32
> test.df = df[-train_index,]         #147 x 32
>
> #Check Multicollinearity
vif(df[,-32])
      Variables      VIF
1          yr.0      Inf
2          yr.1      Inf
3        mnth.1      Inf
4        mnth.2      Inf
5        mnth.3      Inf
6        mnth.4      Inf
7        mnth.5      Inf
8        mnth.6      Inf
9        mnth.7      Inf
10       mnth.8      Inf
11       mnth.9      Inf
12      mnth.10      Inf
13      mnth.11      Inf
14      mnth.12      Inf
15        day.1      Inf
16        day.2      Inf
17        day.3      Inf
18    weekday.0      Inf
19    weekday.1      Inf
20    weekday.2      Inf
21    weekday.3      Inf
22    weekday.4      Inf
23    weekday.5      Inf
24    weekday.6      Inf
25 weathersit.1      Inf
26 weathersit.2      Inf
27 weathersit.3      Inf
28       dteday 1.010204
29        atemp 6.049203
30          hum 2.294781
31    windspeed 1.207595
> vifcor(df[,-32], th = 0.8)
3 variables from the 31 input variables have collinearity problem:
```

```
yr.1 weathersit.2 day.2

After excluding the collinear variables, the linear correlation coefficients ranges betwe
en:
min correlation ( windspeed ~ weekday.3 ):  -0.0001206042
max correlation ( weekday.0 ~ day.1 ):  0.6450846

---------- VIFs of the remained variables --------
      Variables      VIF
1          yr.0 1.049547
2        mnth.1      Inf
3        mnth.2      Inf
4        mnth.3      Inf
5        mnth.4      Inf
6        mnth.5      Inf
7        mnth.6      Inf
8        mnth.7      Inf
9        mnth.8      Inf
10       mnth.9      Inf
11      mnth.10      Inf
12      mnth.11      Inf
13      mnth.12      Inf
14        day.1      Inf
15        day.3 1.106961
16    weekday.0      Inf
17    weekday.1      Inf
18    weekday.2      Inf
19    weekday.3      Inf
20    weekday.4      Inf
21    weekday.5      Inf
22    weekday.6      Inf
23 weathersit.1 1.779943
24 weathersit.3 1.222714
25       dteday 1.010204
26        atemp 6.049203
27          hum 2.294781
28    windspeed 1.207595
> #Output:
> #3 variables from the 31 input variables have collinearity problem: yr.1, weathersit.2,
day.2
> #removing multicollinear variables and redo check:
> df = subset(df, select= -c(yr.1, weathersit.2, day.2))
> train.df = subset(train.df, select= -c(yr.1, weathersit.2, day.2))  #584 x 29
> test.df = subset(test.df, select= -c(yr.1, weathersit.2, day.2))    #147 x 29
> dim(df)  #731 x 29
[1] 731  29
> #Recheck VIFCORR: No variable from the 29 input variables has collinearity problem.
>
> #run regression model
> lr = lm(cnt~., data = train.df)
> #summary of the model
> summary(lr)

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

Residuals:
    Min      1Q  Median      3Q     Max
-3876.2  -387.8    50.8   509.4  2771.2

Coefficients: (3 not defined because of singularities)
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   4430.566    344.461  12.862  < 2e-16 ***
```

```
yr.0          -2113.166     71.121 -29.712  < 2e-16 ***
mnth.1         -825.824    175.718  -4.700 3.29e-06 ***
mnth.2         -716.510    179.767  -3.986 7.62e-05 ***
mnth.3          138.034    174.608   0.791 0.429552
mnth.4          632.261    191.820   3.296 0.001043 **
mnth.5          957.277    209.921   4.560 6.29e-06 ***
mnth.6          673.222    240.603   2.798 0.005319 **
mnth.7          362.334    258.956   1.399 0.162305
mnth.8          644.409    241.596   2.667 0.007868 **
mnth.9         1396.680    213.404   6.545 1.35e-10 ***
mnth.10        1391.067    187.618   7.414 4.56e-13 ***
mnth.11         785.587    172.682   4.549 6.61e-06 ***
mnth.12              NA         NA      NA       NA
day.1             8.810    129.991   0.068 0.945990
day.3          -813.416    212.812  -3.822 0.000147 ***
weekday.0      -424.315    129.802  -3.269 0.001146 **
weekday.1      -165.593    133.805  -1.238 0.216395
weekday.2      -151.960    130.711  -1.163 0.245504
weekday.3       -23.876    130.518  -0.183 0.854920
weekday.4       -54.480    133.389  -0.408 0.683114
weekday.5            NA         NA      NA       NA
weekday.6            NA         NA      NA       NA
weathersit.1    448.480     95.588   4.692 3.41e-06 ***
weathersit.3  -1468.420    232.217  -6.323 5.24e-10 ***
dteday          -10.119      3.989  -2.537 0.011455 *
atemp          4592.019    519.614   8.837  < 2e-16 ***
hum           -1522.767    365.632  -4.165 3.61e-05 ***
windspeed     -2629.300    543.404  -4.839 1.69e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 837.3 on 558 degrees of freedom
Multiple R-squared:  0.8237,  Adjusted R-squared:  0.8158
F-statistic: 104.3 on 25 and 558 DF,  p-value: < 2.2e-16
```

We predict for test set:

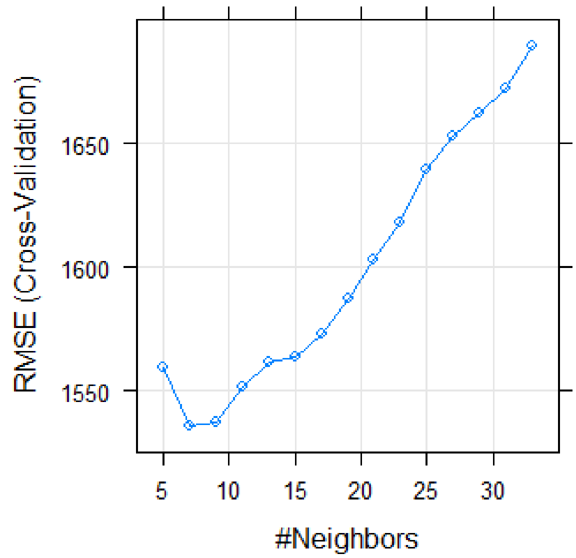
```
predict.lr= predict(lr, test.df[,-29])
```

2.2.5. KNN Implementation

KNN is distance based non-parametric algorithm and it never stores patterns from the training data, but classifies for new test cases based on a similarity measure.

First, we need to check for the best no. of neighbors (k):

```
#To check for best k value:
model <- train(cnt~., data = train, method = "knn",
            trControl = trainControl("cv", number = 10),
            tuneLength = 15)
model$bestTune
#k = 3, 9
plot(model)
```



After checking both methods, it is best to choose k=3 as it gives us the least prediction error.

III. Conclusion

3.1 Model Evaluation:

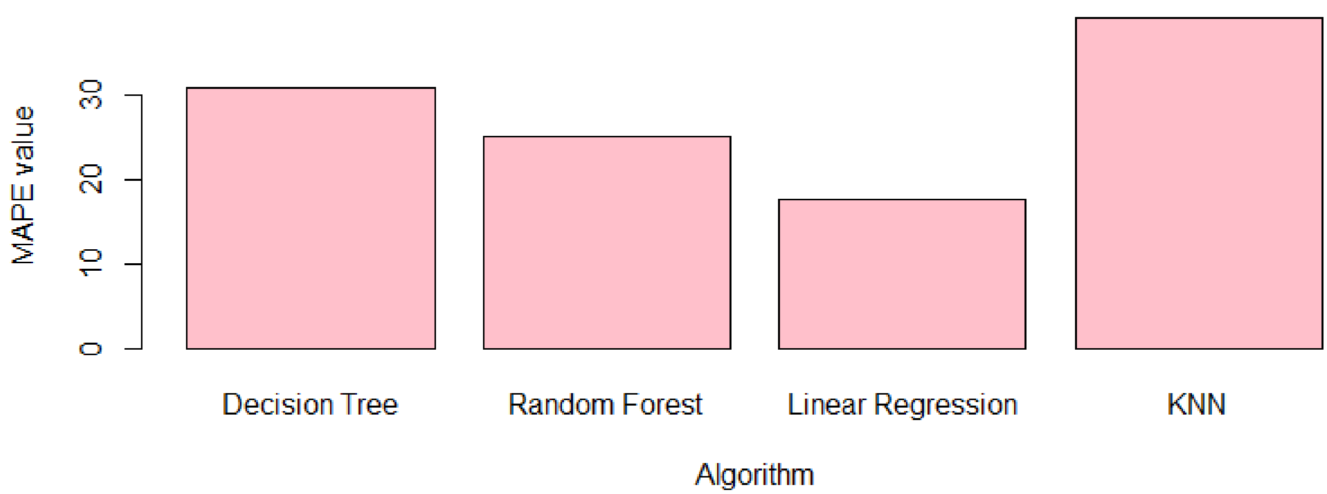
Now that we have a few models for predicting the target variable, we need to decide which one to choose. Several criteria exist for evaluating and comparing models; here we can compare the models by using assessing the 'Predictive Performance' of the models. Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure like RMSE or MAPE.

```
#Error metric for Decision Tree:
postResample(predict.dt,test[,10])
#Output:
#RMSE        Rsquared    MAE
#1036.8218286    0.7105788  768.8217306

#Error metric for Random Forest:
postResample(predict.rf,test[,10])
#Output:
#RMSE        Rsquared       MAE
#778.4675527   0.8507608 576.6110231

#Error metric for Multiple Linear Regression:
postResample(predict.lr,test.df[,29])
#Output:
#RMSE        Rsquared        MAE
#800.2783046   0.8303233 581.4298996
```

```
#Error metric for KNN:
postResample(predict.knn$pred,test.df[,29])
#Output:
#RMSE         Rsquared        MAE
#1392.7631351    0.4544424 1110.0045351

#calculate MAPE
> mape = function(y,yi)
+   {mean(abs((y-yi)/y))*100
+   }
> mape.dt = mape(test[,10],predict.dt)     #30.79%
> mape.rf = mape(test[,10],predict.rf$predict.rf..subset.test..select....c.cnt...)   # 24.9%
> mape.lr = mape(test.df[,29],predict.lr)       #17.5%
> mape.knn = mape(test.df[,29],predict.knn$pred)  #38.98%
```

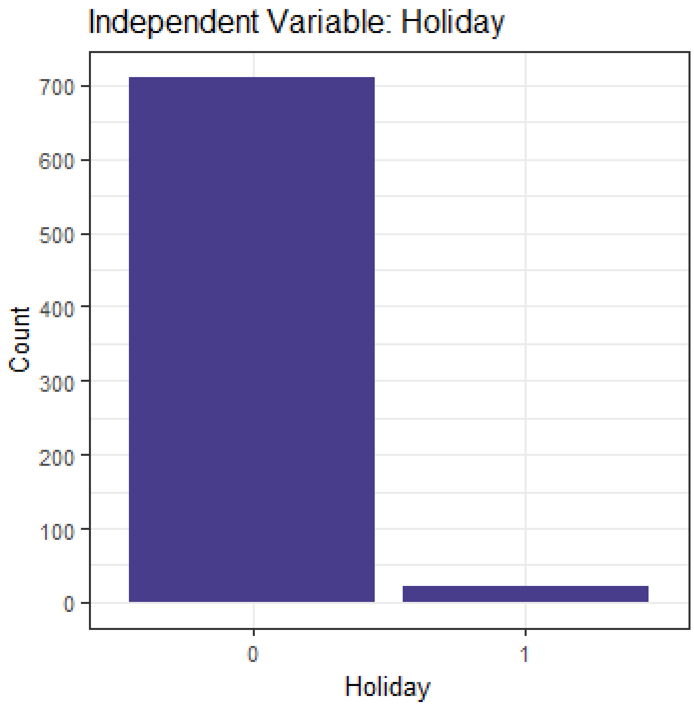
```
>       algorithm MAPE_val
1     Decision Tree 30.79662
2     Random Forest 24.98612
3 Linear Regression 17.55068
4               KNN 38.98097
```

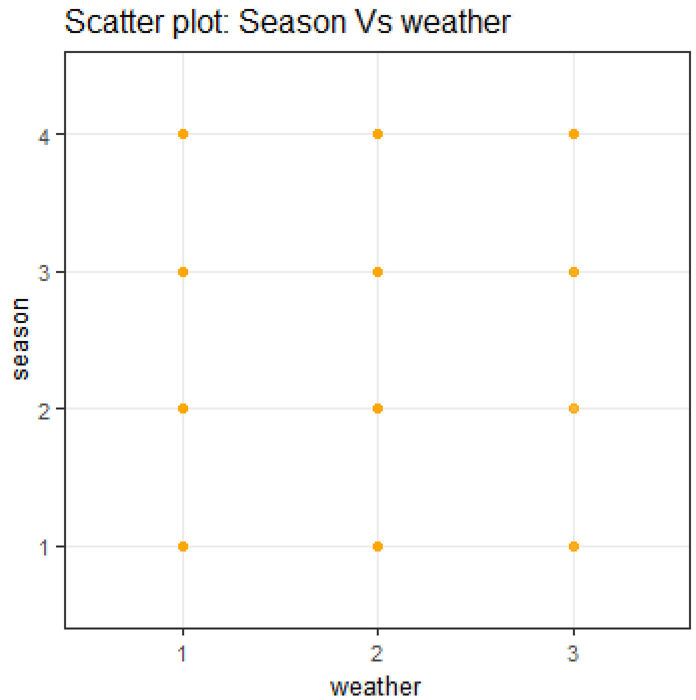
3.2 Final Model Selection:

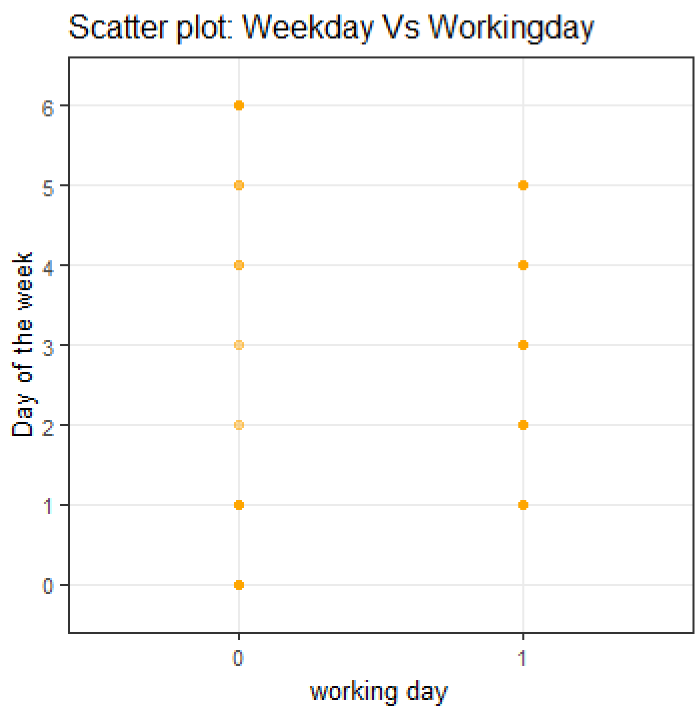


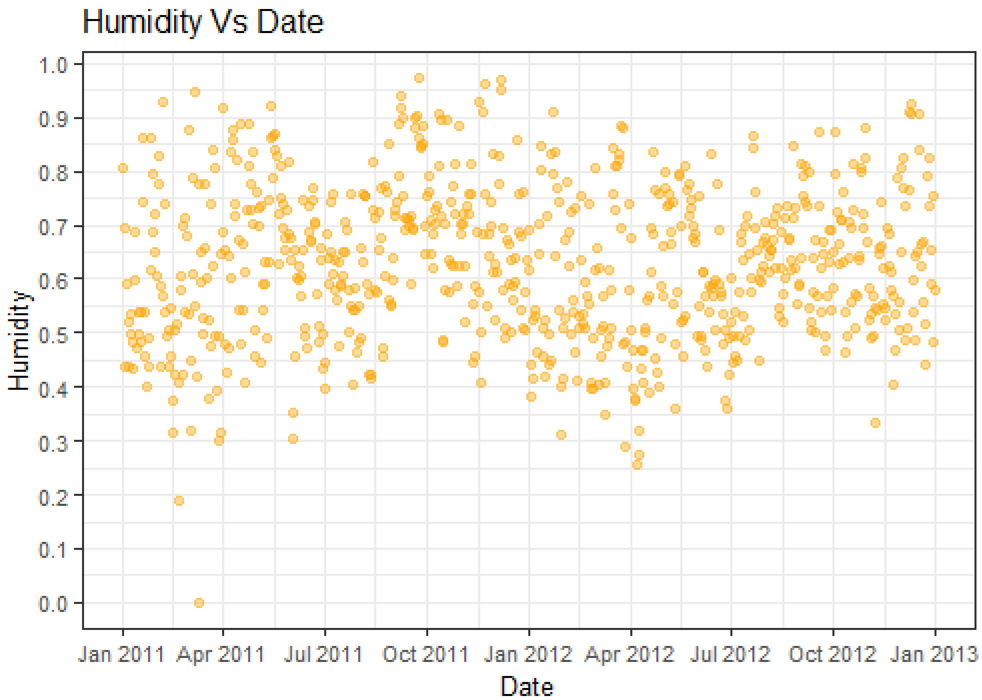
As we can observe that "Multiple Linear Regression" algorithm produces the least error or MAPE (Mean Absolute Percentage Error), we can freeze this algorithm as the model for analysis of new daily data or test cases of Bike Rental count for further years.

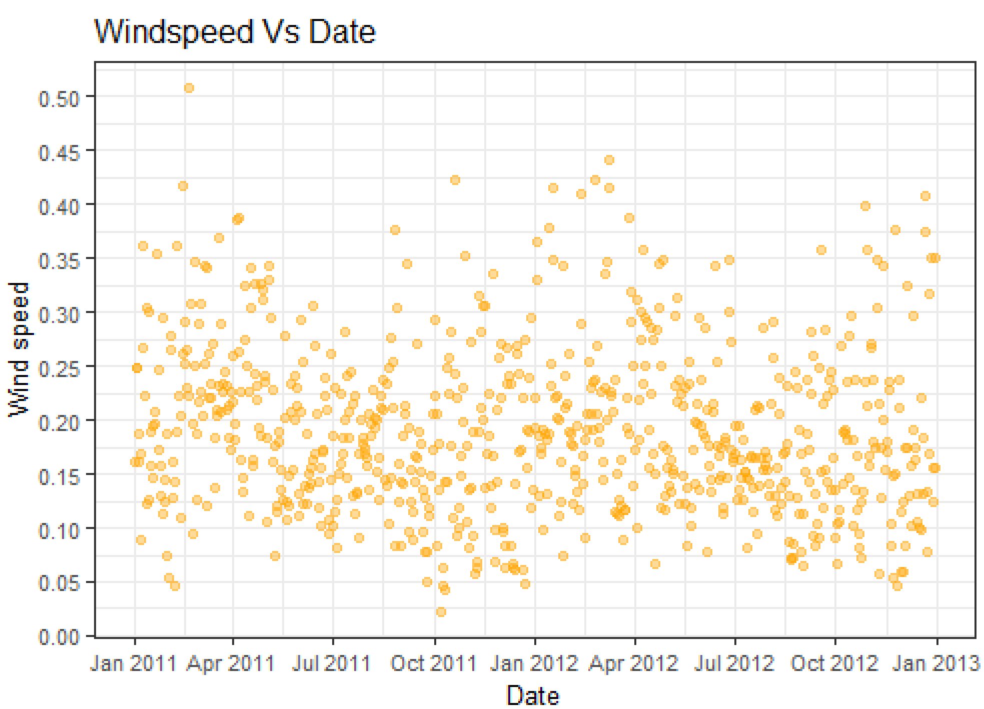
Appendix A: Extra plots

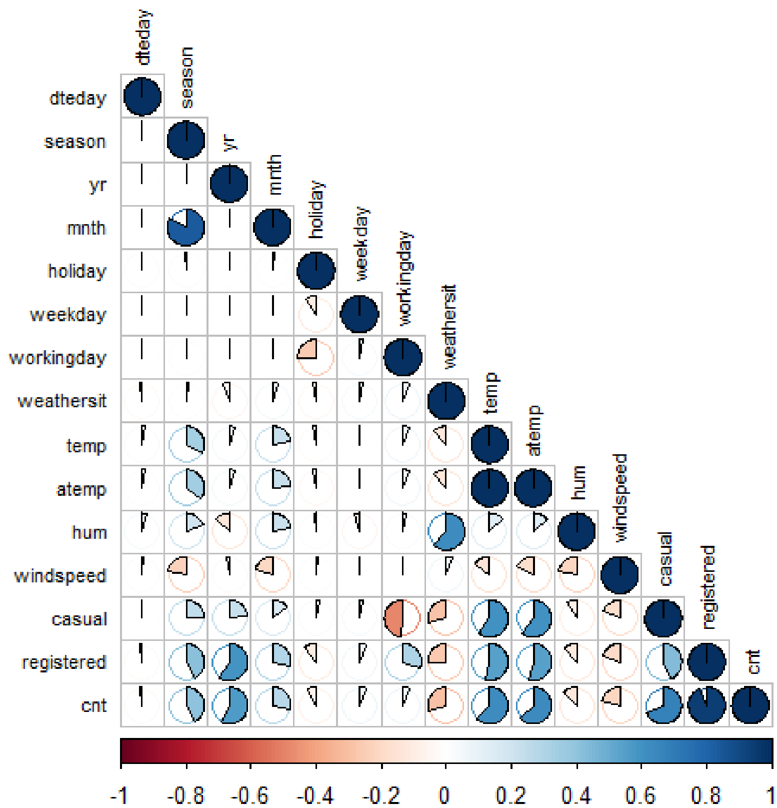




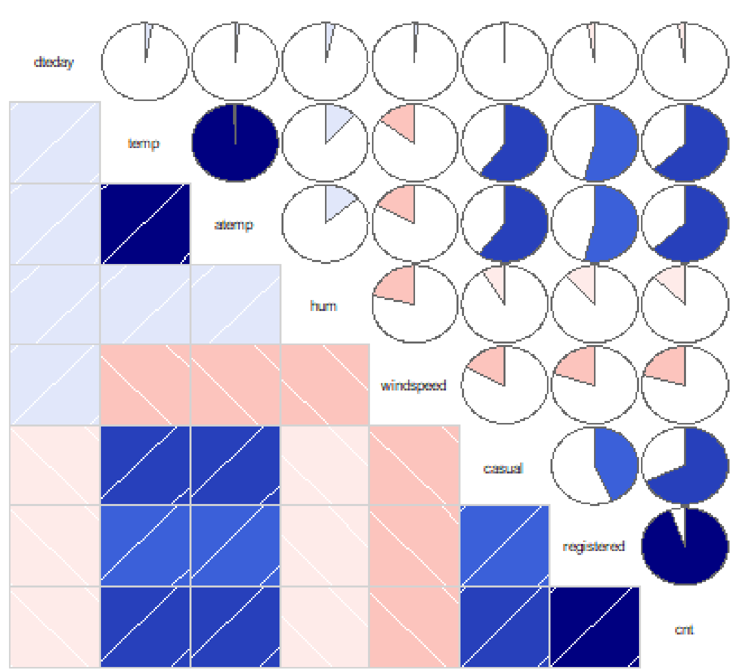



Humidity Vs Date
Humidity
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Jan 2011
Apr 2011
Jul 2011
Oct 2011
Jan 2012
Apr 2012
Jul 2012
Oct 2012
Jan 2013
Date


Windspeed Vs Date
Wind speed
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Jan 2011
Apr 2011
Jul 2011
Oct 2011
Jan 2012
Apr 2012
Jul 2012
Oct 2012
Jan 2013
Date



Correlation Plot



Appendix B: R code

```
#To clear the R environment of any predefined objects
rm(list=ls())
#To set working directory
setwd("F:/DS/edWisor/Project 2")
getwd()

#To load required libraries
library(ggplot2)    # used for ploting
library(dplyr)      # used for data manipulation and joining
library(scales)     # used for "pretty_brakes() function"
library(DMwR)       # used for KNN Imputation
library(outliers)   # used for outlier detection & modification
library(corrgram)   # used for plotting correlation amongst variables
library(corrplot)   # used for plotting correlation amongst variables
library(caret)      # used for various model training
library(lubridate)  # used for handling date format data
library(FNN)        # used for KNN modeling
library(randomForest) # used for Random Forest implementation
library(rpart)      # used for Decision Tree algorithm implementation

#To load the data
data = read.csv("day.csv",header = T, na.strings = c(""," ","NA",NA))

####################Data Exploration######################
str(data)         #"data.frame"
dim(data)           # 731 x 16

###Univariate Analysis###
#col = names(data)
#To find the unique values in each column
#for (i in col) {
#  print(i)
#  print(length(unique(data[,i])))
#}
#Data has 7 categorical variables, 8 numeric variables & one date type variable.
#Target variable is integer type in nature.

###Data Consolidation###
#Convert into Proper data types
#-->ignoring "instant" as it is just like serial number.
data = data[,-1]
#dim(data)      #731 x 15
```

```
#_____Data type conversion_____#
catnames = c("season","yr","mnth","holiday","weekday","workingday","weathersit")   #categorical variables
for (i in catnames) {
  data[,i] = as.factor(data[,i])
}

numnames = c("temp","atemp","hum","windspeed","casual","registered","cnt")        #numerical variables
for (i in numnames) {
  data[,i] = as.numeric(data[,i])
}

data$dteday = as.Date(data$dteday)     #It changed date "02-04-11" to "2011-04-02".


str(data)

###__________________________Graphical analysis________________________________###
#Histogram for Target variable (continuous variable)
ggplot(data, aes_string(x = data$cnt)) +
  geom_histogram(fill="cornsilk", colour = "black") + geom_density() +
  scale_y_continuous(breaks=pretty_breaks(n=10)) +
  scale_x_continuous(breaks=pretty_breaks(n=10))+
  theme_bw() + xlab("Count of total rental bikes") + ylab("Frequency") + ggtitle("Target Variable Histogram") +
  theme(text=element_text(size=10))

#Histogram for Independent Continuous Variables
ggplot(data, aes_string(x = data$temp)) +
  geom_histogram(fill="blue", colour = "black") + geom_density() +
  scale_y_continuous(breaks=pretty_breaks(n=10)) +
  scale_x_continuous(breaks=pretty_breaks(n=10))+
  theme_bw() + xlab("Normalized Temperature (*C)") + ylab("Frequency") +
ggtitle("IndependentVariable:Norm Temperature") +
  theme(text=element_text(size=10))
.
.
.
#And so on. The graphs are plotted and recorded in the project report.


###To extract days from "dteday" and make a new variable
data$day = day(data$dteday)
#As we already have information about the year and month, we have the whole date information & can
remove the "dteday" date type variable as it may not be suitable for modeling.
data[,1] = data[,16]
data[,16] = NULL          #dim = 731 x 15
```

```
col = names(data)

##################______________Missing Value Analysis______________##################
sum(is.na(data))

#There are no missing values for this data set.

####################____________Outlier Analysis____________########################
####Box Plot distribution & outlier check####
str(data)
for(i in 1:length(numnames)){
  assign(paste0("gn",i), ggplot(aes_string(y = (numnames[i]), x = data$cnt), data = subset(data))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "light blue",outlier.shape=18,outlier.size=3, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=numnames[i],x="Bike Rental Count")+
           ggtitle(paste("Box plot for",numnames[i])))
}

#Plotting plots together
gridExtra::grid.arrange(gn1,gn2,gn3,gn4,ncol=4)
gridExtra::grid.arrange(gn5,gn6,gn7,ncol=3)

#To check number of outliers in data (ignoring categorical variables, checked earlier)
out = 0.0
for(i in numnames){
   val = data[,i][data[,i] %in% boxplot.stats(data[,i])$out]
   out = out + length(val)
   print(i)
   print(length(val))
}
out #= 59. Total Outliers in the data set is 59.
#(59/731)*100 = 8.07% of data.

##To test for the best method to find missing values for this dataset
#data[12,12]   #data[12,12] = 0.304627 (actual)
#data[12,12]= NA
#By median method:
#data$windspeed[is.na(data$windspeed)]=median(data$windspeed, na.rm = T)
#data[12,12]  #data[12,12] = 0.180971 (median)

#reupload data
#data[12,12]   #data[12,12] = 0.304627 (actual)
#data[12,12]= NA
```

```
#by mean method:
#data$windspeed[is.na(data$windspeed)]=mean(data$windspeed, na.rm = T)
#data[12,12]  #data[12,12] = 0.1903299 (mean)

#reupload data
#data[12,12]   #data[12,12] = 0.304627 (actual)
#data[12,12]= NA
#By KNN imputation method:
#(KNN takes only numeric inputs)
#for (i in col) {
#  data[,i] = as.numeric(data[,i])
#}
#data= knnImputation(data, k=3)      #For k=5,7,9, the difference was even more than k=3.
#data[12,12]  #data[12,12] = 0.2324425 (KNN)
#We freeze NA imputation by MEDIAN method as it is closest to actual value.

#reupload data
#Converting outliers to NAs
#Select variables with outliers
Out_Var = c('hum','windspeed','casual')  #Variables with outliers

for(i in Out_Var){
  val = data[,i][data[,i] %in% boxplot.stats(data[,i])$out]
  data[,i][data[,i] %in% val] = NA
}
sum(is.na(data)) #To verify

data= knnImputation(data, k=3)

sum(is.na(data)) #To verify

#Confirm again if any outlier exists
out = 0.0
for(i in numnames){
  val = data[,i][data[,i] %in% boxplot.stats(data[,i])$out]
  out= out + length(val)
  print(i)
  print(length(val))
}
out #= 3. Windspeed has 2 outliers & Casual has 1 outlier.

#-->Redo 2 times the imputing using NAs by KNN imputation until 0 outliers.

write.csv(data, 'data_without Outliers.csv', row.names = F)
```

```
#To load the data
#data = read.csv("data_without Outliers.csv",header = T)

############################Feature Selection##############################
#Correlation Plot
corrgram(data, order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot", font.labels = 1)
#cor(x), x must be numeric
#Convert all columns to numeric type
#for (i in col) {
#  data[,i] = as.numeric(data[,i])
#}    #NOTE: This changes all zero factor levels to numeric 1. so, "0" --> 1.
#mat = cor(data)
#corrplot(as.matrix(mat),method= 'pie',type = "lower", tl.col = "black", tl.cex = 0.7)
#If |r|>0.8, those two variables are redundant variables.
#Output: "mnth"-"season", "temp"-"atemp" & "cnt"-"registered" are highly positively correlated.
str(data)
#redo data conversion to proper types
catnames = c("season","yr","mnth","holiday","weekday","workingday","weathersit")   #categorical variables
for (i in catnames) {
  data[,i] = as.factor(data[,i])
}

numnames = c("dteday","temp","atemp","hum","windspeed","casual","registered","cnt")  #numerical
variables
for (i in numnames) {
  data[,i] = as.numeric(data[,i])
}

#######Chi-square Test of Independence (within Categorical Variables)
for(i in catnames){
  for(j in catnames){
    if(i!=j){
     print(names(data[i]))
     print(paste0(" Vs ", names(data[j])))
     print(chisq.test(table(data[,j],data[,i])))
    }
  }
}
#If p-value<0.05 (Reject Null Hypothesis) => variable A depends on variable B.
#If p-value>0.05 (Do Not Reject Null Hypothesis) => Variable A & variable B are independent of each other.
#Output: "workingday"-"holiday","weekday"-"workingday","weekday"-"holiday" & "mnth"-"season depend
on each other significantly.

#######Using Random Forest Algorithm:
```

```
data.rf=randomForest(data$cnt~.,data = data, ntree=1000, keep.forest= F, importance= T)
importance(data.rf,type = 1)
#"holiday" has the least importance.
varImpPlot(data.rf,type = 1)

#######ANOVA test (comparision of Target Vs categorical variables)
anovacat = aov(cnt ~ season + yr + mnth + holiday + workingday + weekday + weathersit , data = data)
summary(anovacat)
#If p-value<0.05 (Reject Null Hypothesis) => Population means are significantly different.
#If p-value>0.05 (Do Not Reject Null Hypothesis) => Population means are not significantly different or are
same.

###################____________Feature Engineering_____________######################
#From Chi-square test, we notice that "working day", "holiday" & "weekday" depend on each other and
intuitively there is a logical connection within them.
#We make a new variable using this connection between the three varibles
#Denote: 1-->weekend, 2--> working day, 3--> holiday

data$day = NA
for (i in 1:nrow(data)){
  if ((data[i,7]=="0") && (data[i,5]=="0")){data[i,16] = 1}              #weekend
  else if ((data[i,7]=="1") && (data[i,5]=="0")){data[i,16] = 2}         #working day
  else if ((data[i,7]=="0") && (data[i,5]=="1")){data[i,16] = 3}         #holiday
  else data[i,16] =NA
}
sum(is.na(data$day)) #= 0, so no anomaly data case where it is working day & holiday both.

###################Dimensional Reduction######################
#Won't remove "dteday" variable as the user count is tracked on each day.
#As we added "day" new variable using "workingday" & "holiday", we can remove them both as "day" holds
the information of both.
data$holiday = data$day
data$day = NULL
colnames(data)[5] = "day"
data$day = as.factor(data$day)     # New variable "day": Factor w/ 3 levels "1","2","3"
#"Season" has multicollinearity problem as well and it is related to "mnth", so we can remove it.
data= subset(data, select= -c(season,workingday,temp,casual,registered))
factor_data = subset(data, select= c(yr,mnth,day,weekday,weathersit))  #5 factor variables
num_data = subset(data, select= c(dteday,atemp,hum,windspeed,cnt))  #5 numerical variables, contains
target variable
dim(data)          # 731 obs. x 10 variables
str(data)
################################Feature Scaling##################################
#All continuous variables are already normalised in this data set.
```

```
rm(list= ls()[!(ls() %in% c('data','factor_data','num_data'))])

##############################Sampling#################################
set.seed(777)

sample.index = sample(nrow(data), 0.8*nrow(data), replace = F)   #80% data -->Train set, 20%--> Test set
train = data[sample.index,]
test = data[-sample.index,]
dim(train)    # 584 x 11
dim(test)     # 147 x 11

##################################Model Development###################################
#As the target variable is of numeric type, this is a regression problem.
######1.Decision Tree######
#Decision trees can handle both categorical and numerical variables at the same time as features.
dt=rpart(cnt~.,data = train,method= "anova")
summary(dt)

#Predict for new test cases
predict.dt=predict(dt,test[,-10])

#Error metric:
postResample(predict.dt,test[,10])
#Output:
#RMSE         Rsquared     MAE
#1036.8218286    0.7105788  768.8217306

#calculate MAPE
mape = function(y,yi)
  {mean(abs((y-yi)/y))*100
  }
mape.dt = mape(test[,10],predict.dt)     #30.79%

library(mltools)
rmsle(predict.dt,test[,10])    #0.3665

#######2.Random Forest Algorithm#######
rf = randomForest(cnt~., train, importance = TRUE, ntree = 500)
summary(rf)
#Predict for test case:
predict.rf <- data.frame(predict(rf, subset(test, select = -c(cnt))))
#Error metric:
postResample(predict.rf,test[,10])
#Output:
```

```
#RMSE         Rsquared       MAE
#778.4675527   0.8507608 576.6110231

mape.rf = mape(test[,10],predict.rf$predict.rf..subset.test..select....c.cnt...)   # 24.9%

########3.Multiple Linear Regression########

#creating dummy variables for categorical data
library(dummies)
factor_new = dummy.data.frame(factor_data, sep = ".")   #731 x 27

#sampling#
df = cbind(factor_new, num_data)
#for (i in 1:ncol(df)) {
#  df[,i] = as.numeric(df[,i])
#}
str(df)          # 731 X 32

set.seed(123)
train_index = sample(1:nrow(df), 0.8*nrow(df))
train.df = df[train_index,]        #584 x 32
test.df = df[-train_index,]        #147 x 32

#Check Multicollinearity
library(usdm)
vif(df[,-32])
vifcor(df[,-32], th = 0.8)
#Output:
#3 variables from the 31 input variables have collinearity problem: yr.1, weathersit.2, day.2
#removing multicollinear variables and redo check:
df = subset(df, select= -c(yr.1, weathersit.2, day.2))
train.df = subset(train.df, select= -c(yr.1, weathersit.2, day.2))  #584 x 29
test.df = subset(test.df, select= -c(yr.1, weathersit.2, day.2))    #147 x 29
dim(df)  #731 x 29
#Recheck VIFCORR: No variable from the 29 input variables has collinearity problem.

#run regression model
lr = lm(cnt~., data = train.df)
#summary of the model
summary(lr)

#Predict for test case:
predict.lr= predict(lr, test.df[,-29])
#Error metric:
postResample(predict.lr,test.df[,29])
```

```
#Output:
#RMSE       Rsquared      MAE
#800.2783046   0.8303233 581.4298996

mape.lr = mape(test.df[,29],predict.lr)      #17.5%

##############4.KNN Implementation##############
#To check for best k value:
model <- train(cnt~., data = train, method = "knn",
            trControl = trainControl("cv", number = 10),
            tuneLength = 15)
model$bestTune
#k = 2 , 7
plot(model)

#K=3:
predict.knn = knn.reg(train = train.df[,-29],test = test.df[,-29],train.df$cnt, k= 3)
print(predict.knn)

#Error metric:
postResample(predict.knn$pred,test.df[,29])
#Output:
#RMSE         Rsquared      MAE
#1392.7631351    0.4544424 1110.0045351

mape.knn = mape(test.df[,29],predict.knn$pred)  #38.98%

#K=5:
#predict.knn = knn.reg(train = train.df[,-29],test = test.df[,-29],train.df$cnt, k= 5)
#print(predict.knn)
#Error metric:
#mape(test.df[,29],predict.knn$pred)
#Output:
#mape
#45.26592 %
#postResample(predict.knn$pred,test.df[,29])
#RMSE         Rsquared      MAE
#1450.9419952    0.4484269 1169.3782313

#K=7:
#predict.knn = knn.reg(train = train.df[,-29],test = test.df[,-29],train.df$cnt, k= 7)
#print(predict.knn)
#Error metric:
#mape(test.df[,29],predict.knn$pred)
#Output:
```

```
#mape
#47.63637 %
#postResample(predict.knn$pred,test.df[,29])
#RMSE           Rsquared        MAE
#1456.0507716    0.4983456 1171.8736638
######And so on, done upto k = 11.

#A new dataframe to store results
algorithm <- c('Decision Tree','Random Forest','Linear Regression','KNN')
MAPE_val <- c(mape.dt,mape.rf,mape.lr,mape.knn)
results <- data.frame(algorithm, MAPE_val)
print(results)
barplot(results$MAPE_val, width = 1, names.arg = results$algorithm,
        ylab="MAPE value", xlab = "Algorithm",col='pink')

##Thus, we find the "Multiple Linear Regression Algorithm" gives us the best result with the least MAPE for this dataset.
```

Appendix C: Python code

```
#Set working directory
import os
os.chdir("F:/DS/edWisor/Project 2")
os.getcwd()
```

Load libraries

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from random import randrange, uniform
from scipy.stats import chi2_contingency
from ggplot import *
```

In [ ]:

```
from fancyimpute import KNN
```

In [ ]:

```
import datetime as dt
```

In [ ]:

```
#Load the data
data = pd.read_csv("day.csv")
```

Data exploration

In [ ]:

```
data.shape
```

In [ ]:

```
data.head(10)
```

In [ ]:

```
type(data)
```

In [ ]:

```
data.info()
```

In [ ]:

```
#Missing Value Analysis
#Check for missing value
data.isnull().sum()
#No missing values in the dataset
```

In [ ]:

```
#remove "instant" variable as it is just like serial number & doesn't predict
data = data.drop(['instant'], axis=1)
```

In [ ]:

```
data.shape
```

In [ ]:

```
#extracting day number from 'dteday' variable
data['dteday'].apply(str)
data['dteday'] = pd.to_datetime(data['dteday'])
data['dteday'] = pd.DatetimeIndex(data['dteday']).day
#removing 'dteday' variable
```

In [ ]:

```
data.head(20)
```

In [ ]:

```
#save numeric & categorical names
numnames = ["dteday","temp","atemp","hum","windspeed","casual","registered","cnt"]
catnames = ["season","yr","mnth","holiday","weekday","workingday","weathersit"]
data.shape
```

In [ ]:

```
for i in catnames:
   data[i] = data[i].astype('object')
for i in numnames:
   data[i] = data[i].astype('float')
```

In [ ]:

```
data.dtypes
```

Outlier analysis

In [ ]:

```
#Plot boxplot to visualize Outliers
%matplotlib inline
plt.boxplot(data['windspeed'])
```

In [ ]:

```
#Detect and delete outliers from data
```

```
for i in numnames:
    print(i)
    q75, q25 = np.percentile(data.loc[:,i], [75 ,25])
    iqr = q75 - q25

    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print(min)
    print(max)

    #Remove the outliers
    data = data.drop(data[data.loc[:,i] < min].index)
    data = data.drop(data[data.loc[:,i] > max].index)

    #data.loc[data[i] < min,:i] = np.nan
    #data.loc[data[i] > max,:i] = np.nan

#Calculate missing value
#missing_val = pd.DataFrame(data.isnull().sum())
#Impute with KNN
#data = pd.DataFrame(KNN(21).fit_transform(data), columns = data.columns)
```

In [ ]:

```
data.shape        #55 rows deleted
```

In [ ]:

```
data.isnull().sum()
```

Feature Selection

In [ ]:

```
##Correlation analysis
#Correlation plot
df_corr = data.loc[:,numnames]
#Set the width and height of the plot
f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix
corr = df_corr.corr()

#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
          square=True, ax=ax)
```

In [ ]:

```
#Chisquare test of independence
#loop for chi square values
for i in catnames:
   print(i)
   chi2, p, dof, ex = chi2_contingency(pd.crosstab(data['cnt'], data[i]))
   print(p)
```

In [ ]:

```
#New Categorical Variable containing the data of "workingday" & "holiday"
#Denote: 1-->weekend, 2--> working day, 3--> holiday
data.loc[(data['workingday'] == 0) & (data['holiday'] == 0), 'day'] = '1'
data.loc[(data['workingday'] == 1) & (data['holiday'] == 0), 'day'] = '2'
data.loc[(data['workingday'] == 0) & (data['holiday'] == 1), 'day'] = '3'
```

In [ ]:

```
data = data.drop(["workingday","holiday","temp","casual","registered"], axis=1)
```

In [ ]:

```
data.head(10)
```

In [ ]:

```
df = data[['dteday','mnth','yr','season','weekday','day','weathersit','atemp','hum','windspeed','cnt']]
```

In [ ]:

```
df.head(10)
```

In [ ]:

```
################################Feature Scaling##################################
#All continuous variables are already normalised in this data set.


numnames = ["dteday","atemp","hum","windspeed"]        #not including "cnt" target variable
catnames = ["mnth","yr","season","weekday","day","weathersit"]
```

Model Development

In [ ]:

```
#Data Sampling
nrow= len(df.index)
train, test = train_test_split(df, test_size = 0.2)
```

In [ ]:

```
train.shape     #540 x 11
test.shape      #136 x 11
```

In [ ]:

```
#####Decision Tree Algortithm
from sklearn.tree import DecisionTreeRegressor
fit_dt= DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:10],train.iloc[:,10])
```

In [ ]:

```
fit_dt
```

In [ ]:

```
predict_dt= fit_dt.predict(test.iloc[:,0:10])
```

In [ ]:

```
#Calculate RMSE
def RMSE(actual, pred):
    return np.sqrt(((pred - actual) ** 2).mean())

RMSE(test.iloc[:,10],predict_dt)
#output = 1162.84440171958
```

In [ ]:

```
######Random Forest Algorithm
from sklearn.ensemble import RandomForestRegressor
fit_rf = RandomForestRegressor(n_estimators = 100, random_state = 99).fit(train.iloc[:,0:10],train.iloc[:,10])
```

In [ ]:

```
fit_rf
```

In [ ]:

```
predict_rf= fit_rf.predict(test.iloc[:,0:10])
```

In [ ]:

```
RMSE(test.iloc[:,10],predict_rf)
#output = 765.0407919968172
```

In [ ]:

```
######Multiple Linear Regression
import statsmodels.api as sm
#Creat dataframe with all numerical variables
df.lr = df[['cnt','dteday','atemp','hum','windspeed']]
#create dummies for categorical variables
for i in catnames:
   temp = pd.get_dummies(df[i],prefix = i)
   df.lr = df.lr.join(temp)
```

In [ ]:

```
df.lr.shape                    #676 x 36
```

In [ ]:

```
#split data into train-test sets
s = np.random.rand(len(df.lr))<0.8
train.lr = df.lr[s]       #80%
test.lr = df.lr[~s]       #20%
```

In [ ]:

```
train.lr.shape           #564 x 36
test.lr.shape            #112 x 36
```

In [ ]:

```
#Build MLR model
fit_lr = sm.OLS(train.lr.iloc[:,0],train.lr.iloc[:,1:35]).fit()
fit_lr.summary()
```

In [ ]:

```
predict_lr = fit_lr.predict(test.lr.iloc[:,1:35])
```

In [ ]:

```
RMSE(test.lr.iloc[:,0],predict_lr)
#output = 713.1957640471251
```

In [ ]:

```
######KNN Implementation
from sklearn import neighbors
rmse_val = []          #to store rmse values for different k
for K in range(30):
   K = K+1
```

```
    fit_knn = neighbors.KNeighborsRegressor(n_neighbors = K)

    fit_knn.fit(train.iloc[:,0:10], train.iloc[:,10]) #fit the model

    predict_knn = fit_knn.predict(test.iloc[:,0:10]) #make prediction on test set
    error = RMSE(test.iloc[:,10] , predict_knn) #calculate rmse
    rmse_val.append(error) #store rmse values
    print('RMSE value for k= ' , K , 'is:', error)
```

In [ ]:

```
#plotting the rmse values against k values
curve = pd.DataFrame(rmse_val)
curve.plot()
#K=2 is the value of neighbors for least RMSE.
```

In [ ]:

```
#For K=12:
fit_knn = neighbors.KNeighborsRegressor(n_neighbors = 2)
fit_knn.fit(train.iloc[:,0:10], train.iloc[:,10]) #fit the model
predict_knn = fit_knn.predict(test.iloc[:,0:10]) #make prediction on test set
RMSE(test.iloc[:,10] , predict_knn)
#output = 1209.595772142617
```

In [ ]:

```
#Thus, we find the "Multiple Linear Regression Algorithm" gives us the best result with the least RMSE for this dataset.
```

References

Mitchell, T. (1997). Machine Learning. McGraw Hill. p. 2

Brieman, Friedman, Olshen and Stone, Classification and Regression Trees, 1984