Predicting Bike Rental Count

POOJA JOSHI

Contents

Chapter 1

Introduction

1.1 Problem Statement

The objective of this Project is to Predict bike rental count on daily based on the environmental and seasonal settings.

1.2 Data

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

- instant: Record indexdteday: Dateseason:
- Season (1:springer, 2:summer, 3:fall, 4:winter)
- yr: Year (0: 2011, 1:2012)
- mnth: Month (1 to 12)hr: Hour (0 to 23)
- holiday: weather day is holiday or not (extracted fromHoliday Schedule)
- weekday: Day of the week
- workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
- weathersit: (extracted fromFreemeteo)
    - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
    - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain +
    - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp: Normalized temperature in Celsius. The values are derived via(t-t_min)/(t_max-t_min),t_min=-8, t_max=+39 (only in hourly scale)
- atemp: Normalized feeling temperature in Celsius. The values are derived via(t-t_min)/(t_max-t_min), t_min=-16, t_max=+50 (only in hourly scale)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

Chapter 2

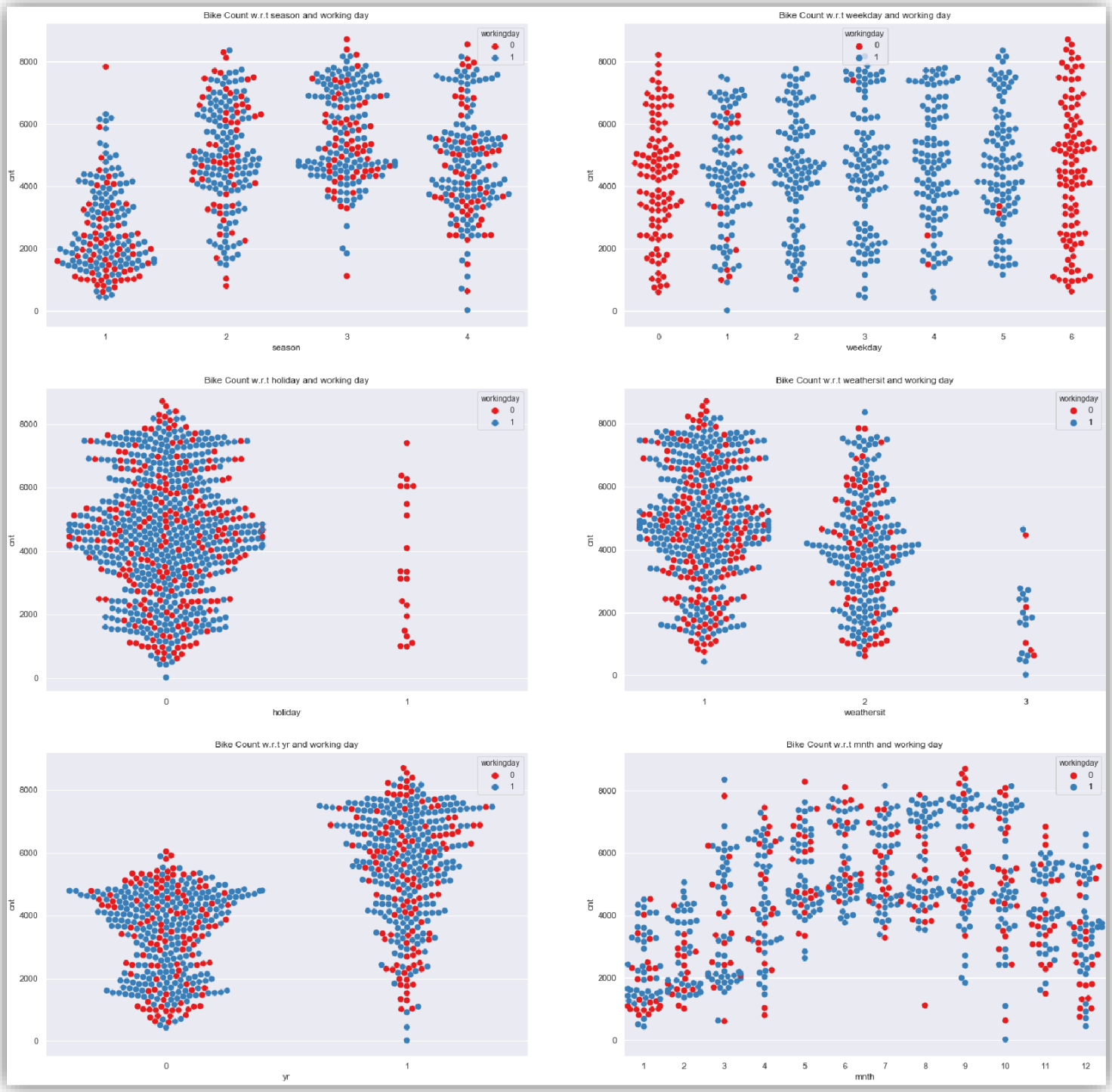
Methodology

2.1 Pre-Processing

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

Getting feel of data via visualization:

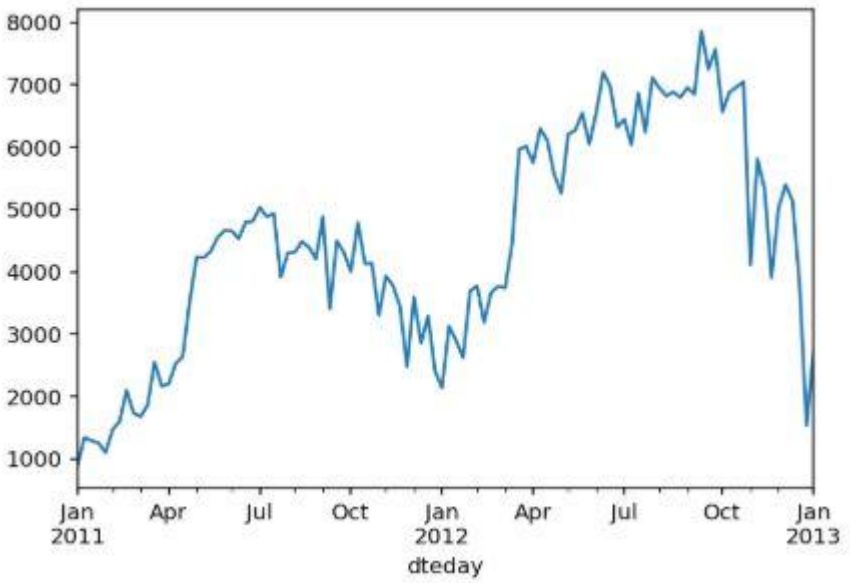
Bee Swarmplots because no binning Bias unlike histogram:



Time Series Visualization:

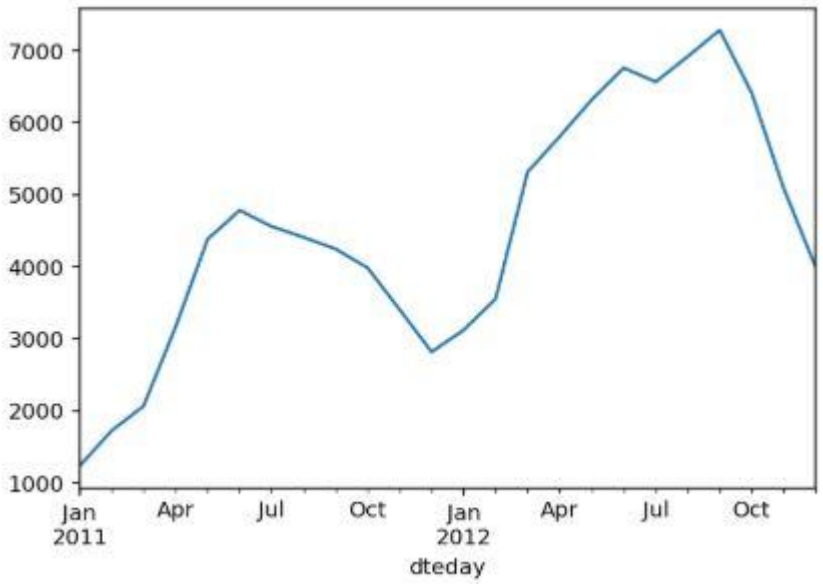
Count according to week of the data:

```
%matplotlib inline
df.cnt.resample('W').mean().plot()
```



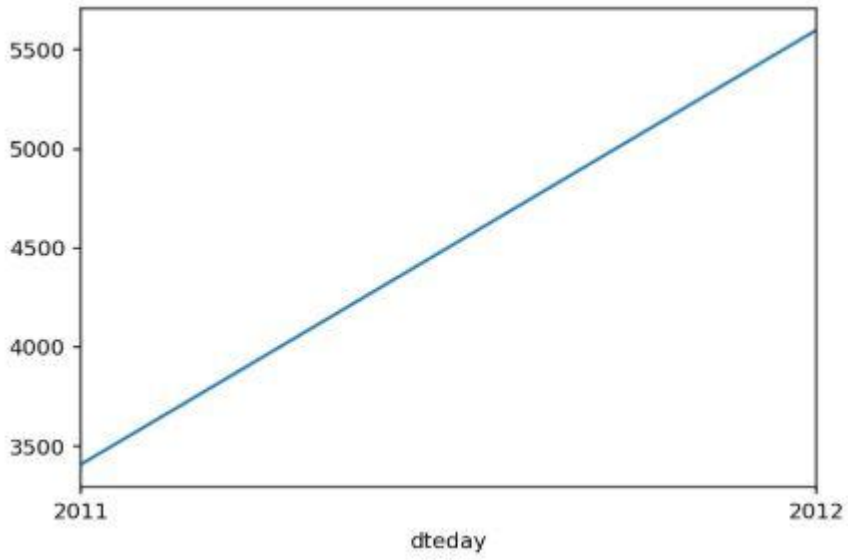
Count according to month of the data:-

```
%matplotlib inline
df.cnt.resample('M').mean().plot()
```

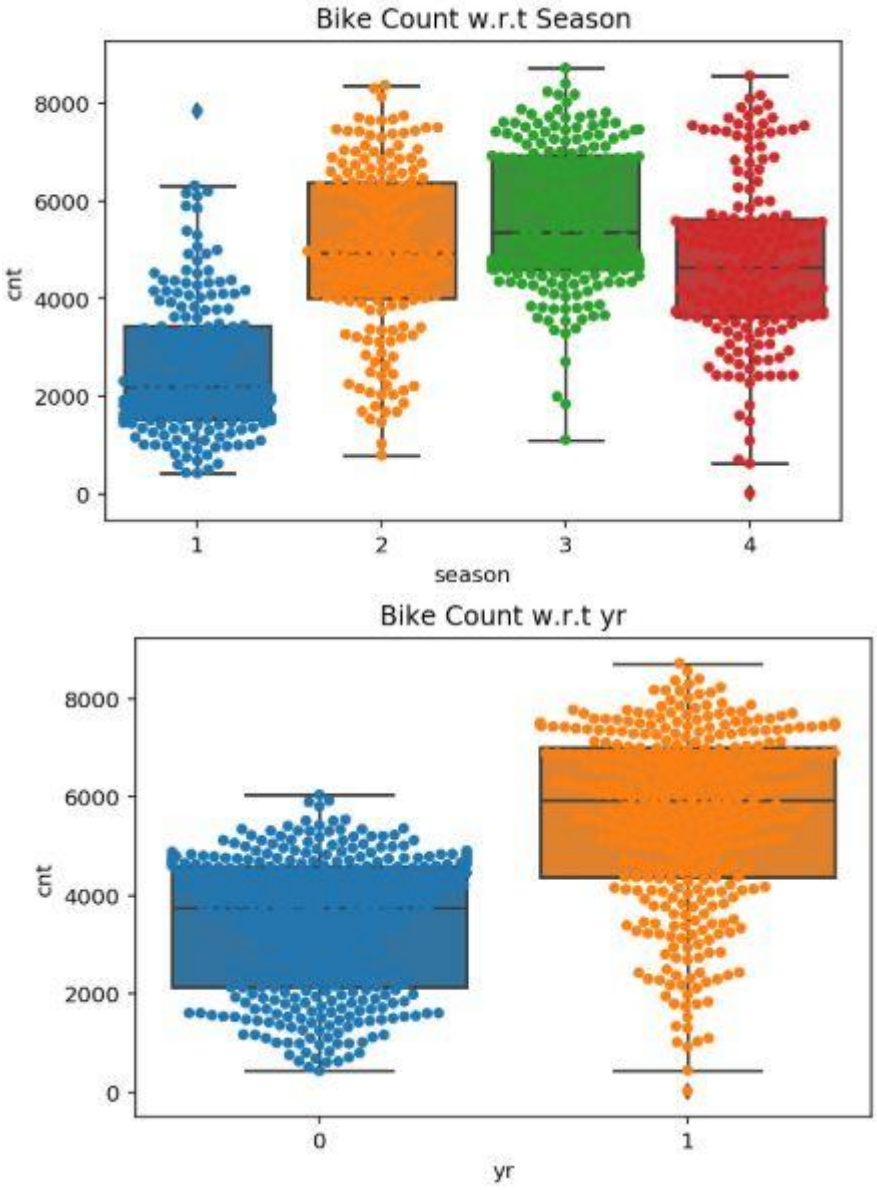


Count according to year of the data:-

```
%matplotlib inline
df.cnt.resample('Y').mean().plot()
```



Swarmplot and Boxplot:-



2.1.1 Missing value Analysis

In this step we look for missing values in the dataset like empty row column cell which was left after removing special characters and punctuation marks.

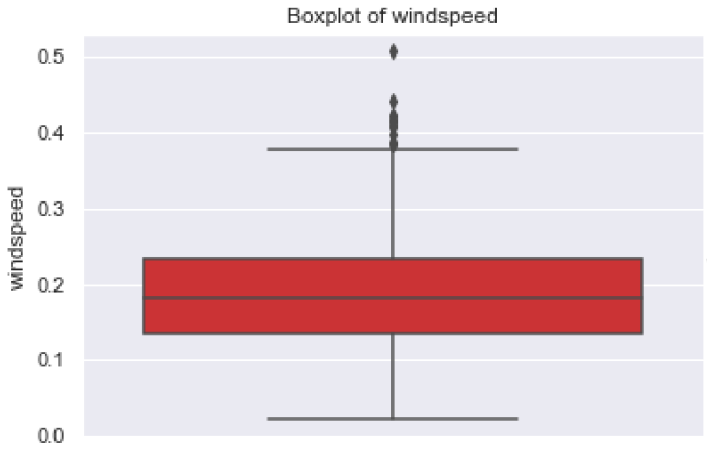
Some missing values are in form of NA. missing values left behind after outlier analysis; missing values can be in any form. Unfortunately, in this dataset we haven't found any missing values. Therefore, we will continue to next step.

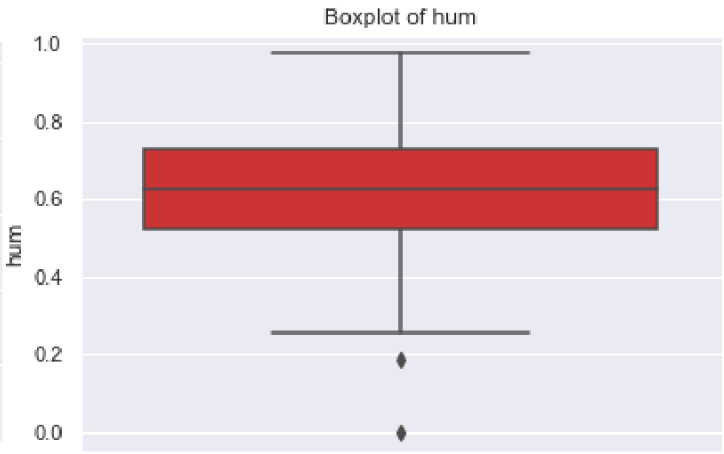
2.1.2 Outlier Analysis

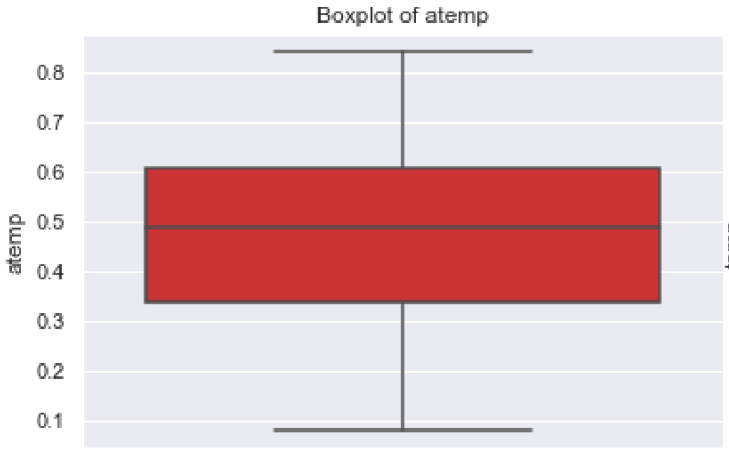
We look for outlier in the dataset by plotting Boxplots. There are outliers present in the data. Now we have removed these outliers. This is how we done,

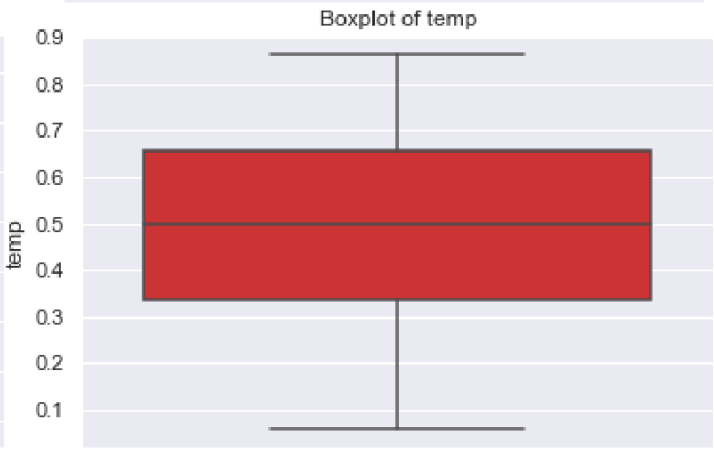
I. We replaced them with Nan values or we can say created missing values.
II. Then we imputed those missing values with KNN method.
III. We tried three methods to impute the missing value: mean, median, KNN. But KNN method outperformed mean and median methods.
IV. We checked the performance of each method by checking Standard Deviation of that variable which has outliers before imputation and after imputation.

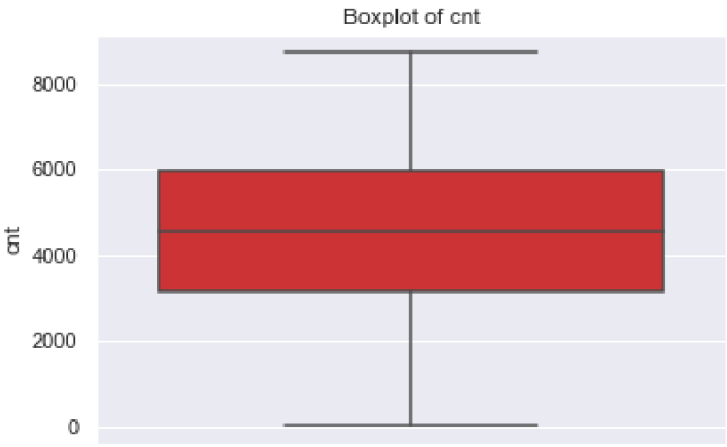
Univariate Boxplots: Boxplots for all Numerical Variables also for target variable



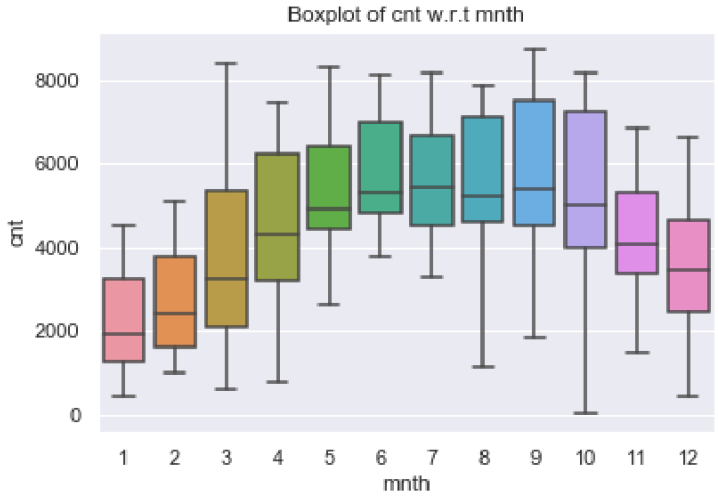


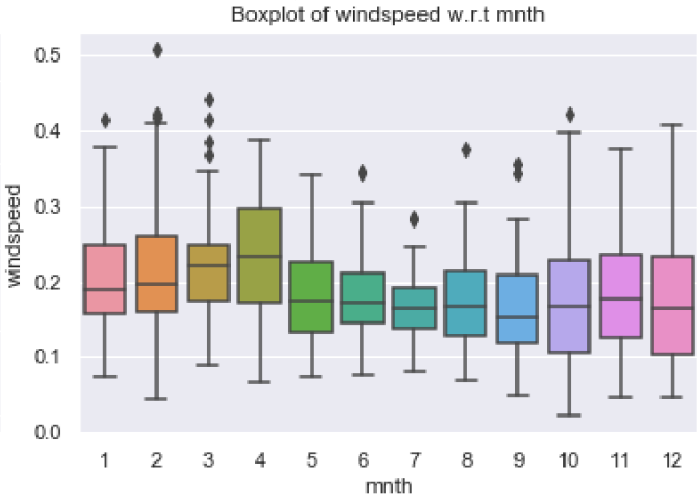


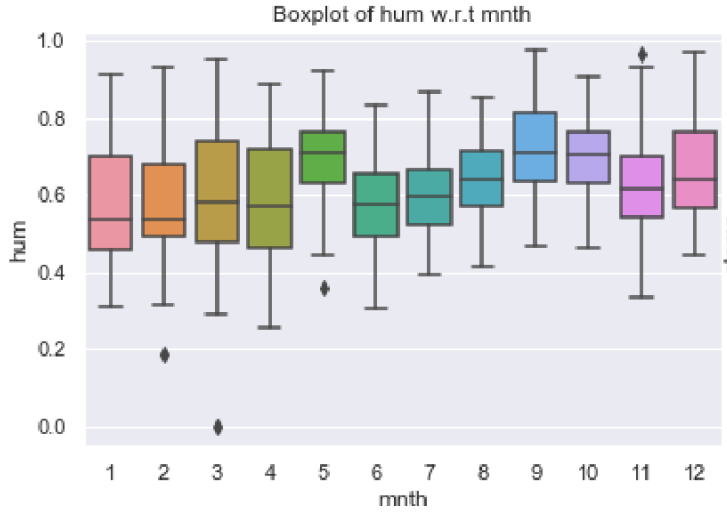


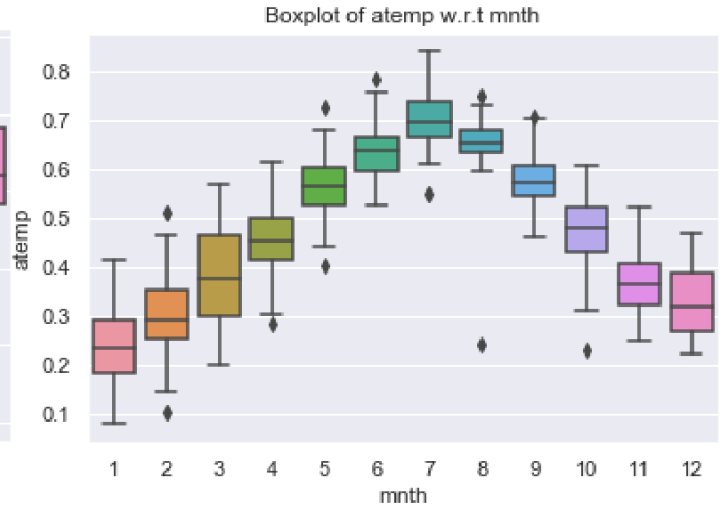


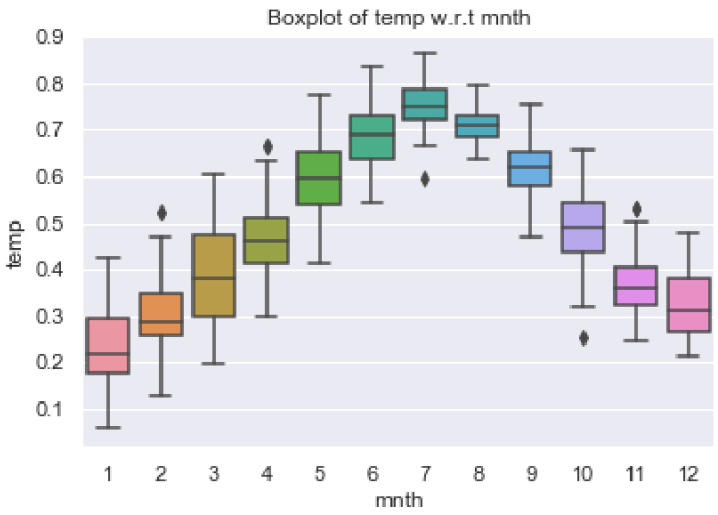
Bivariate Boxplots: Boxplots for all Numerical Variables Vs all Categorical Variables

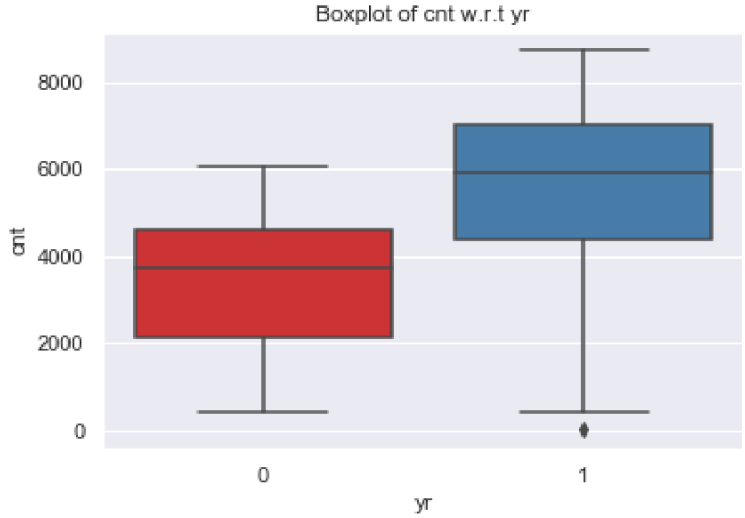


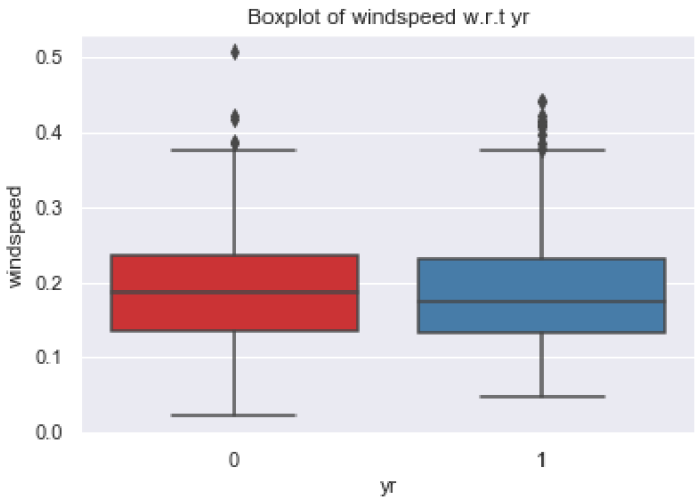


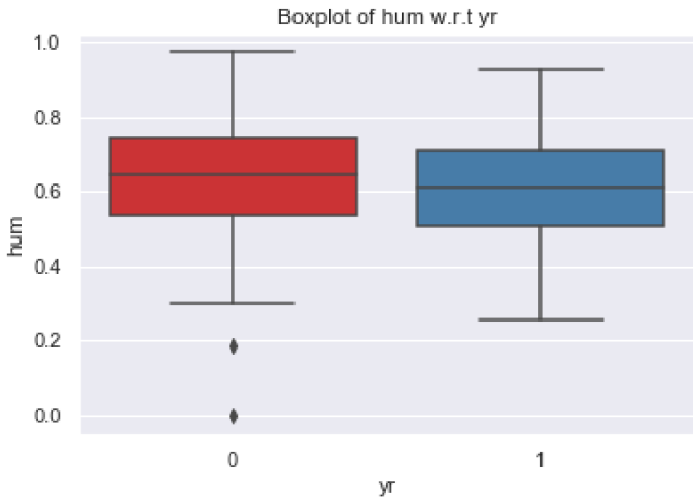


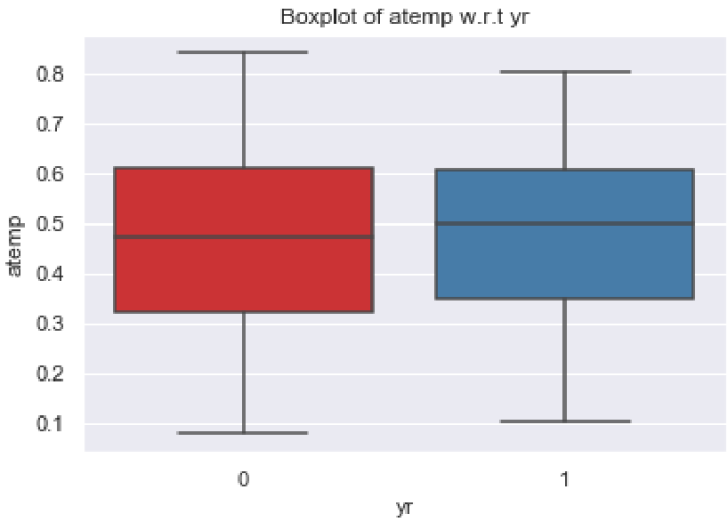


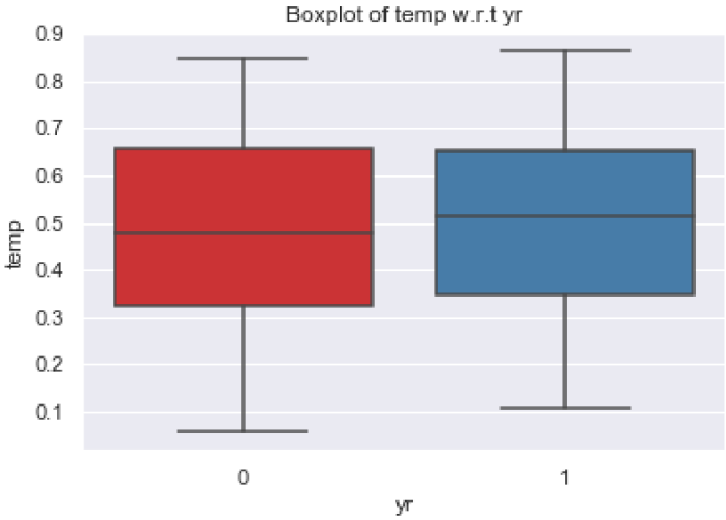


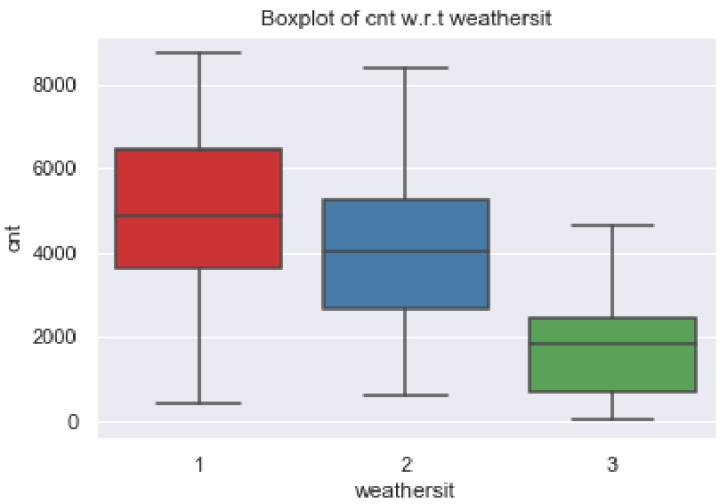


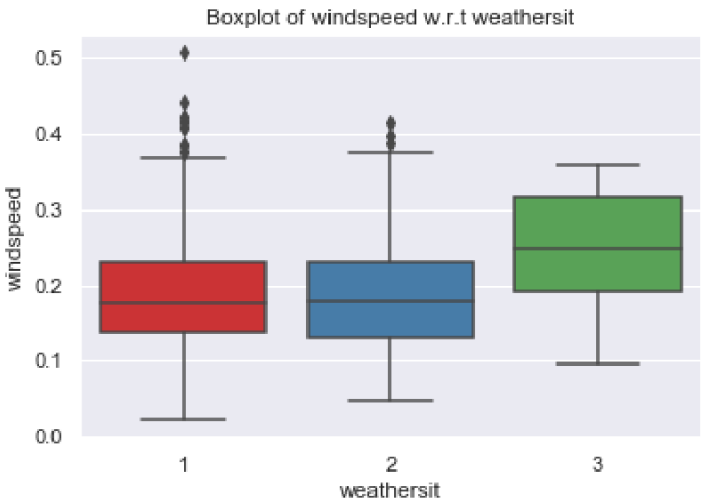

Boxplot of windspeed w.r.t yr
windspeed
0.0
0.1
0.2
0.3
0.4
0.5
0
1
yr

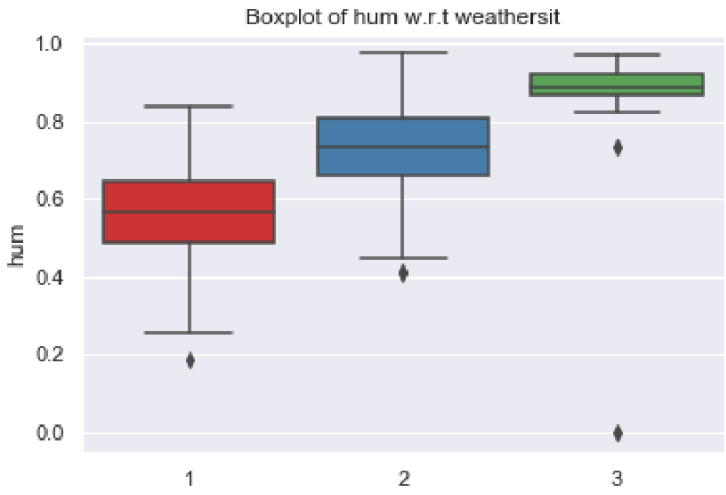

Boxplot of hum w.r.t yr
hum
0.0
0.2
0.4
0.6
0.8
1.0
0
1
yr

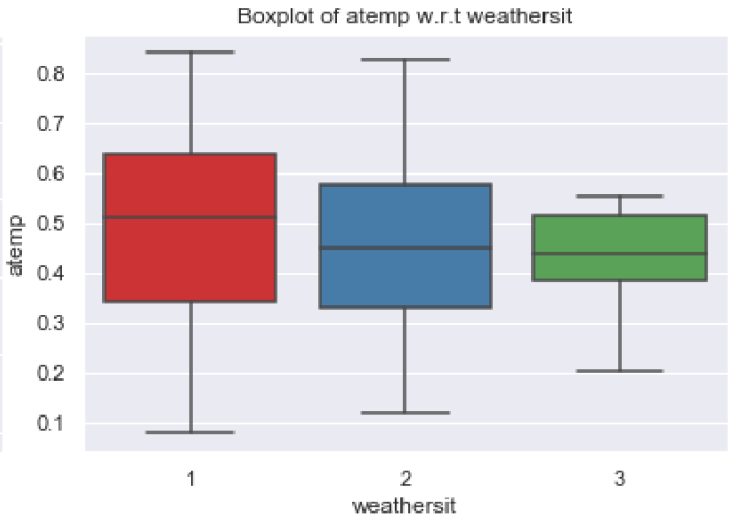

Boxplot of atemp w.r.t yr
atemp
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
yr

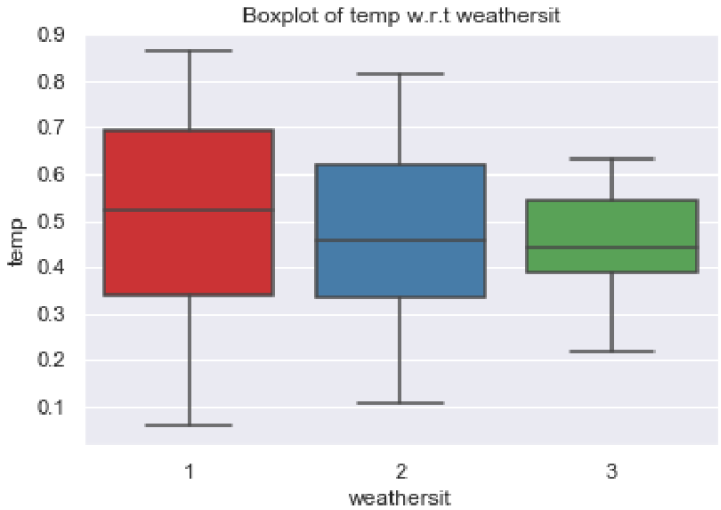

Boxplot of temp w.r.t yr
temp
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
1
yr

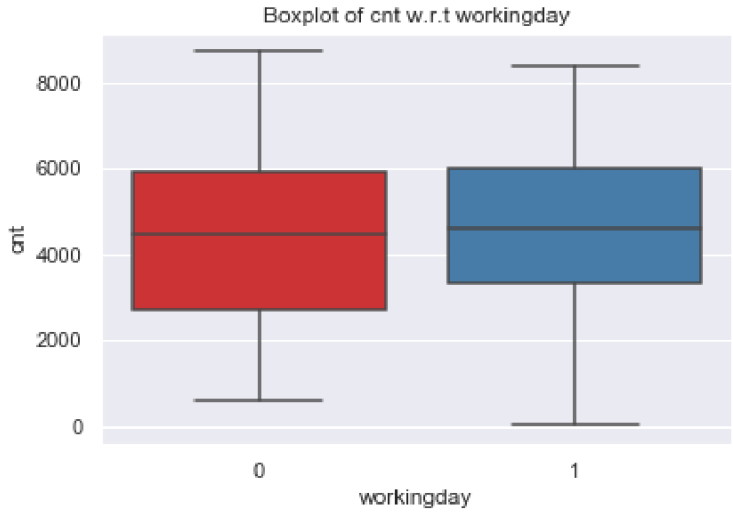

Boxplot of cnt w.r.t weathersit
cnt
0
2000
4000
6000
8000
1
2
3
weathersit

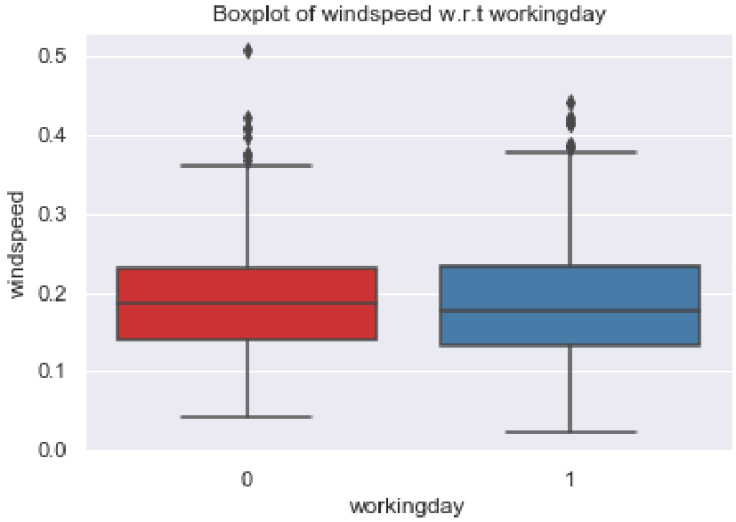

Boxplot of windspeed w.r.t weathersit
windspeed
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
weathersit

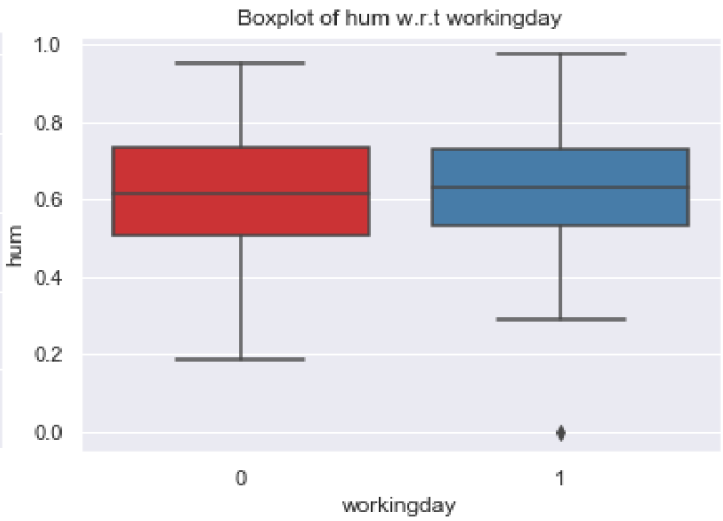

Boxplot of hum w.r.t weathersit
hum
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
weathersit

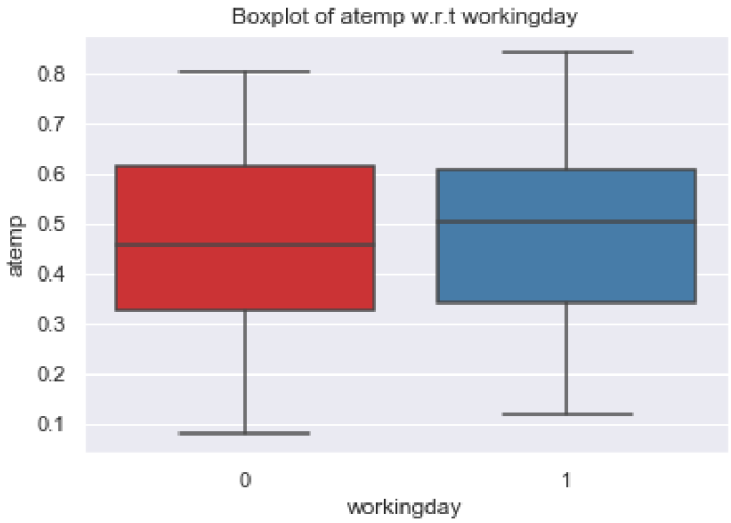

Boxplot of atemp w.r.t weathersit
atemp
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1
2
3
weathersit

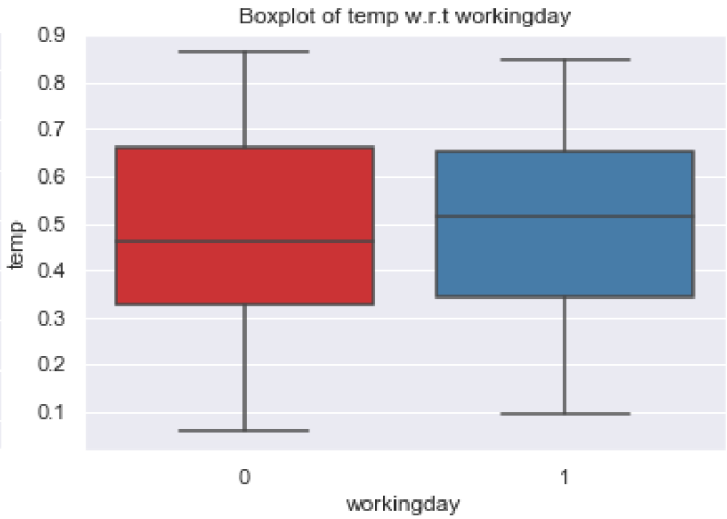

Boxplot of temp w.r.t weathersit
temp
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1
2
3
weathersit

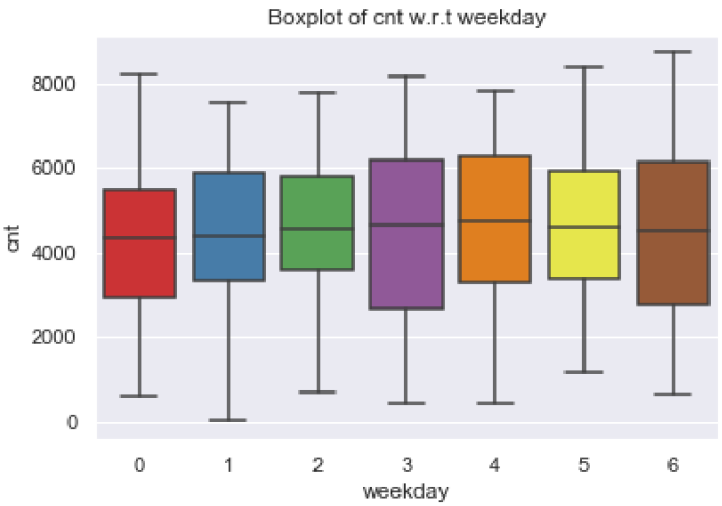

Boxplot of cnt w.r.t workingday
cnt
8000
6000
4000
2000
0
0
1
workingday

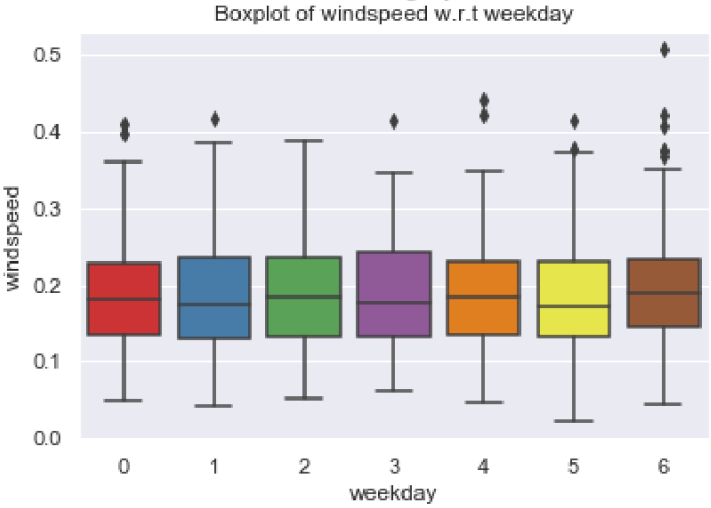

Boxplot of windspeed w.r.t workingday
windspeed
0.5
0.4
0.3
0.2
0.1
0.0
0
1
workingday

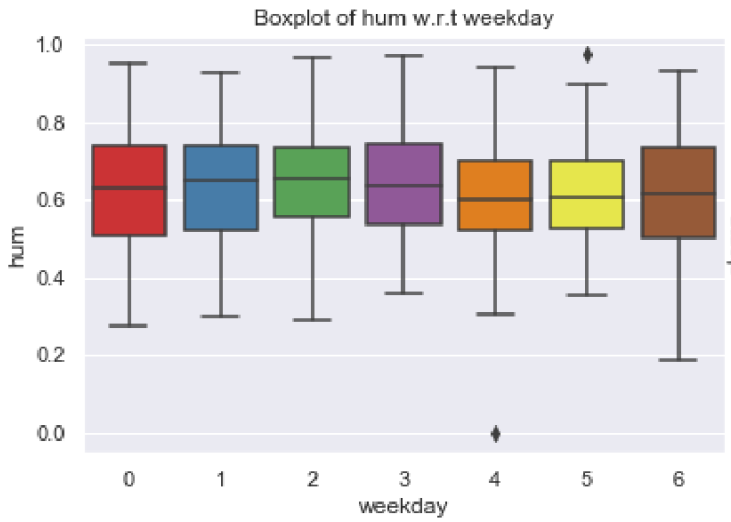

Boxplot of hum w.r.t workingday
hum
1.0
0.8
0.6
0.4
0.2
0.0
0
1
workingday

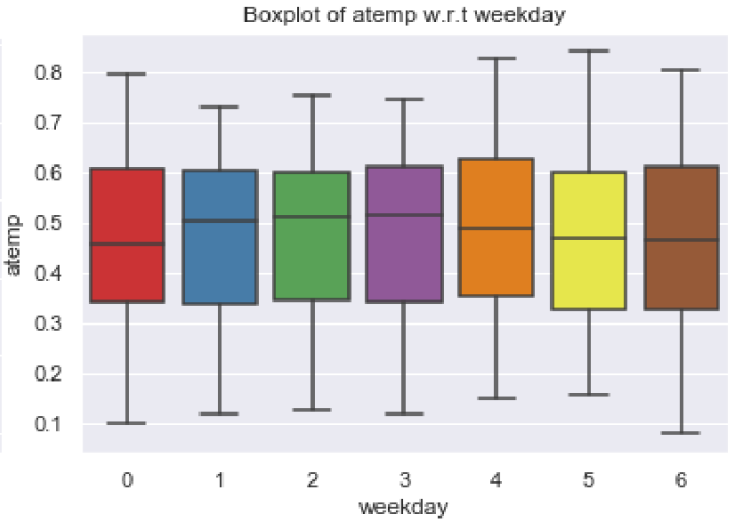

Boxplot of atemp w.r.t workingday
atemp
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
workingday

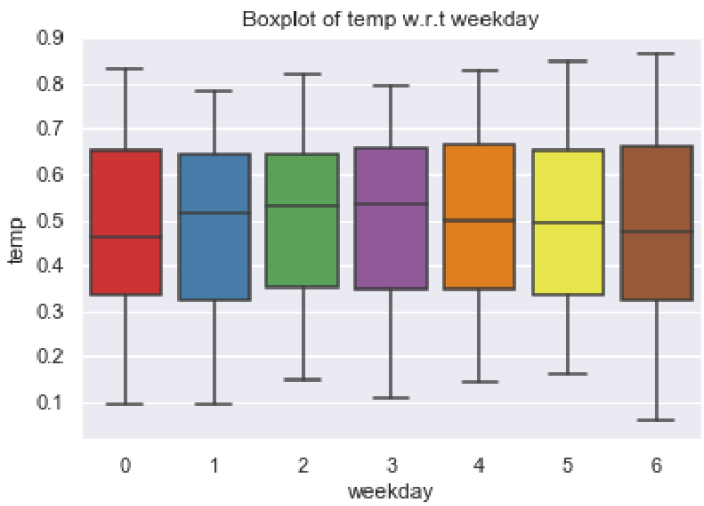

Boxplot of temp w.r.t workingday
temp
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
workingday

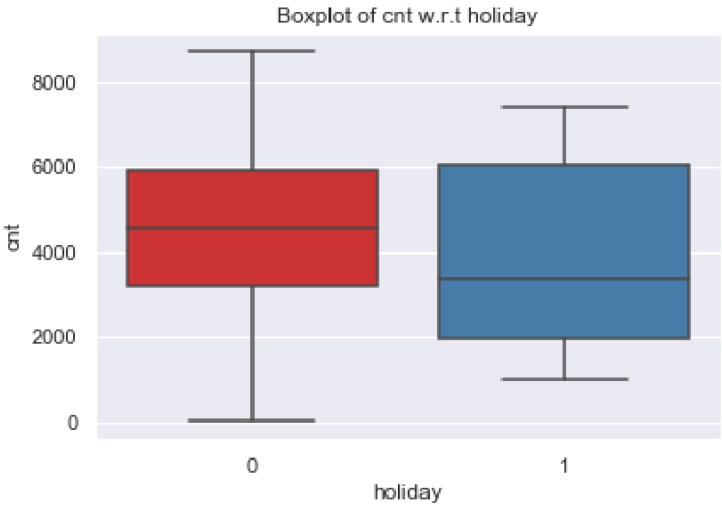

Boxplot of cnt w.r.t weekday
cnt
8000
6000
4000
2000
0
0
1
2
3
4
5
6
weekday

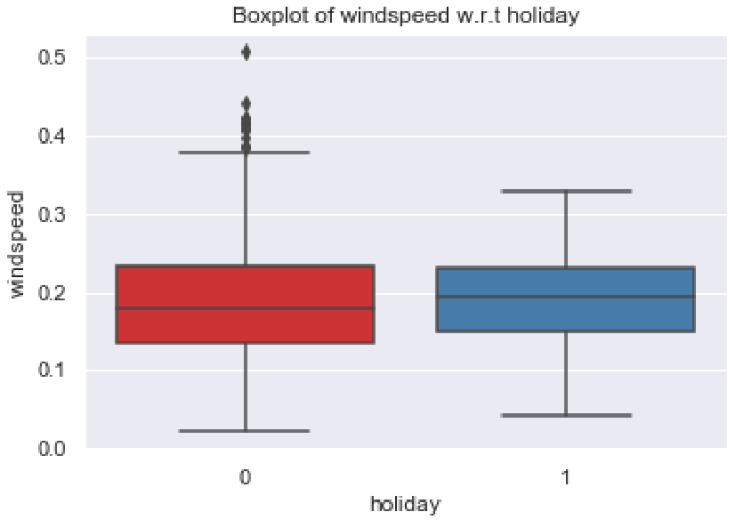

Boxplot of windspeed w.r.t weekday
windspeed
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
5
6
weekday

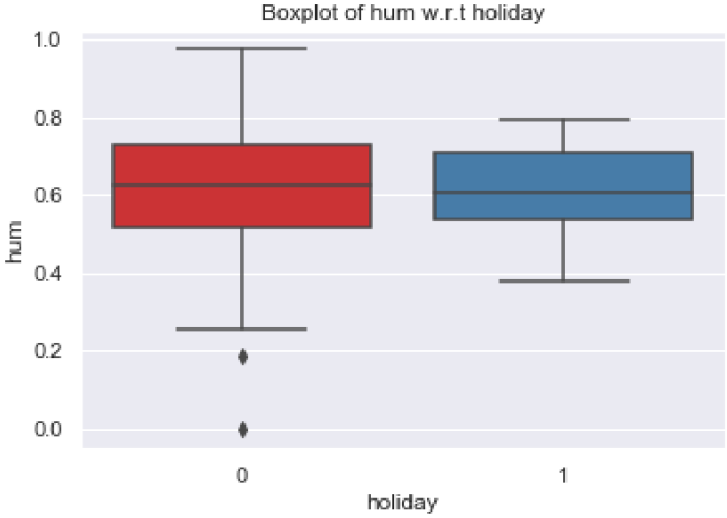

Boxplot of hum w.r.t weekday
hum
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
weekday

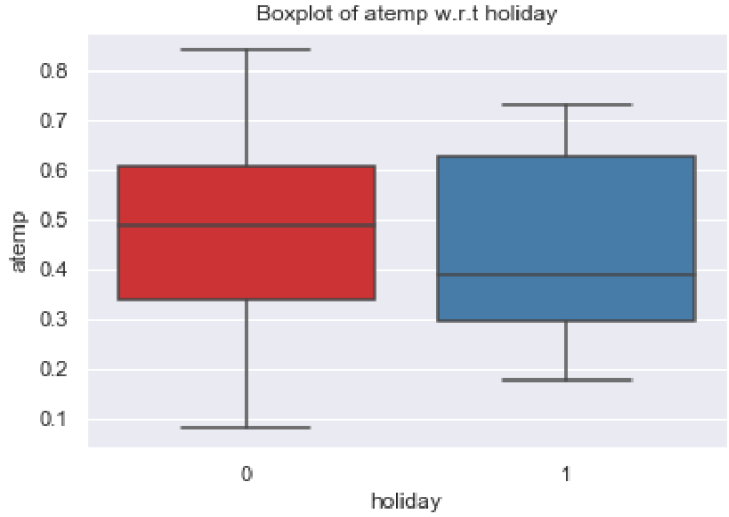

Boxplot of atemp w.r.t weekday
atemp
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
weekday

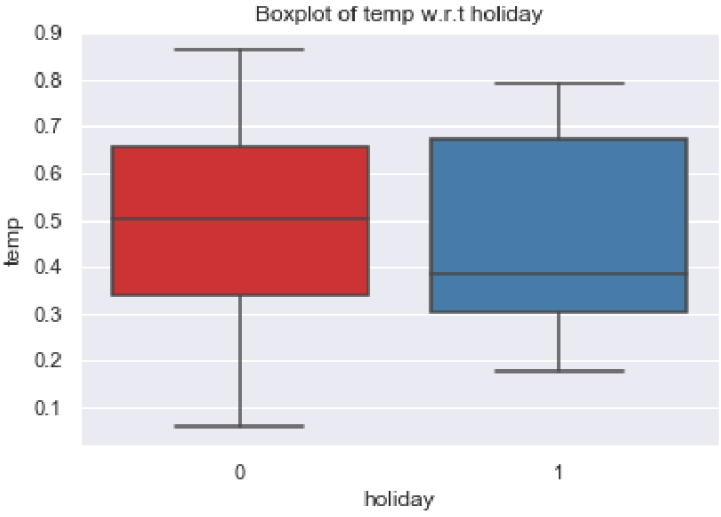

Boxplot of temp w.r.t weekday
temp
weekday
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6

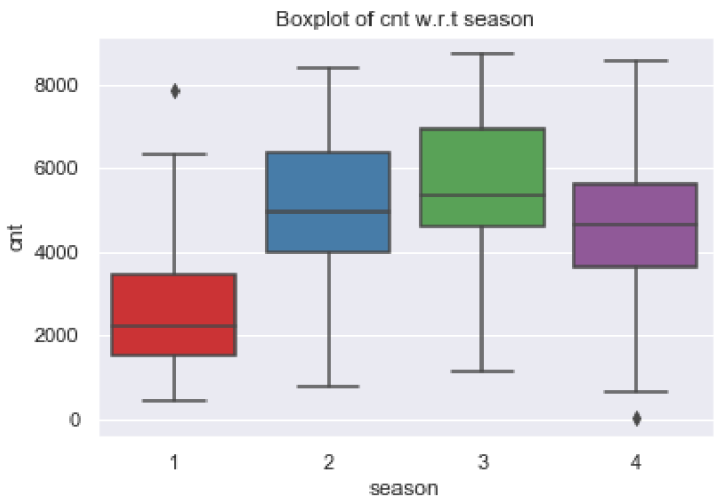

Boxplot of cnt w.r.t holiday
cnt
holiday
8000
6000
4000
2000
0
0
1

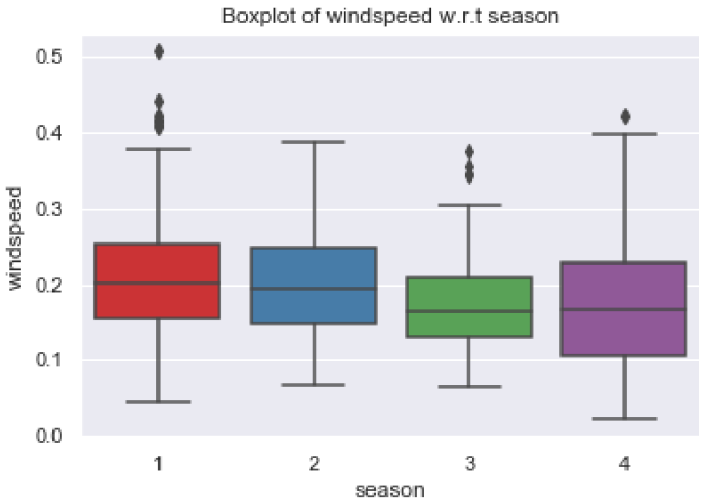

Boxplot of windspeed w.r.t holiday
windspeed
holiday
0.5
0.4
0.3
0.2
0.1
0.0
0
1

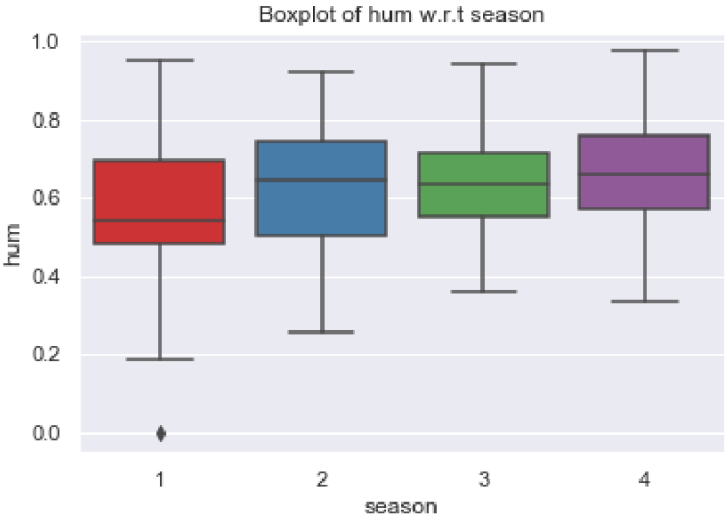

Boxplot of hum w.r.t holiday
hum
holiday
1.0
0.8
0.6
0.4
0.2
0.0
0
1

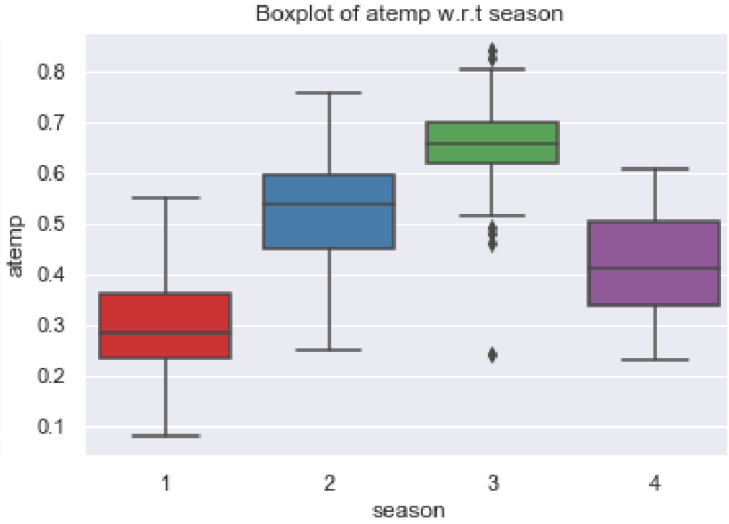

Boxplot of atemp w.r.t holiday
atemp
holiday
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1

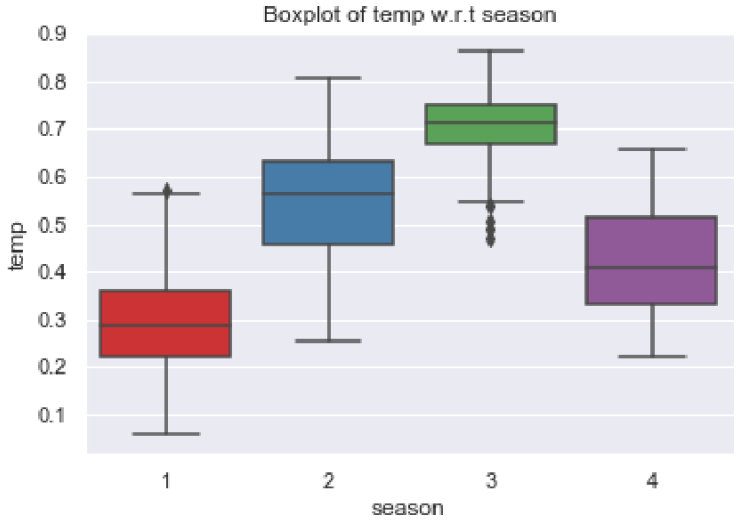

Boxplot of temp w.r.t holiday
temp
holiday
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1











From above Boxplots we see that only ‘hum’ and ‘windspeed’ and ‘casual’ have outliers in them.

‘hum’ has 2 outliers and ‘windspeed’ has 13 outliers and ‘casual’ has 44.

```
[1] "hum"
[1] 2
[1] "windspeed"
[1] 13
[1] "casual"
[1] 44
```

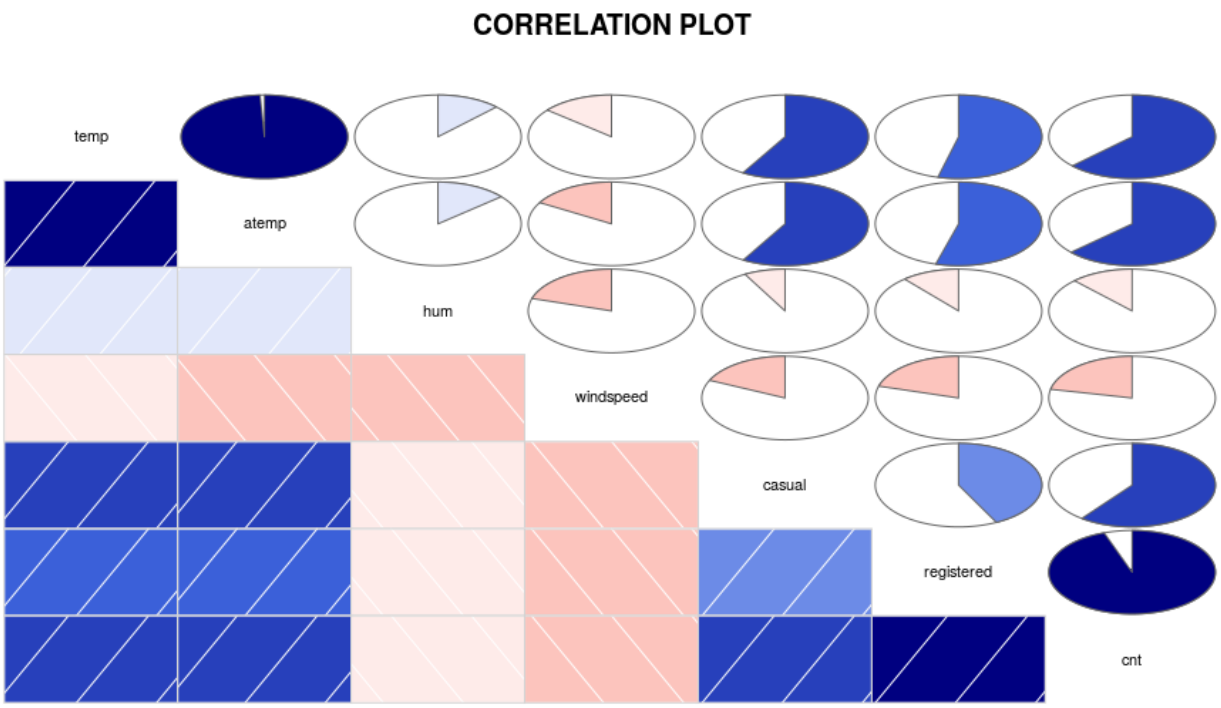
2.1.3 Feature Selection

In this step we would allow only to pass relevant features to further steps. We remove irrelevant features from the dataset. We do this by understanding the domain knowledge of our features like we look for features which will not be helpful in predict the target variables. In this dataset we have to predict the Count based on environmental and seasonal features, features which excludes this list is – instant.
Further below are some types of test involved for feature selection:

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

- 'temp' and 'atemp' are very highly correlated with each other.
- Similarly, 'registered' and 'cnt' are highly correlated with each other.
- We also came to know that--'cnt'='casual'+'registered' .

Correlation Plot:



2 **Chi-Square test of independence** – Unlike correlation analysis we will filter out only categorical variables and pass it to Chi-Square test. Chi-square test compares 2 categorical variables in a contingency table to see if they are related or not.

I. Assumption for chi-square test: Dependency between Independent variable and dependent variable should be high and there should be no dependency among independent variables.

II. Before proceeding to calculate chi-square statistic, we do the hypothesis testing:

Null hypothesis: 2 variables are independent.

Alternate hypothesis: 2 variables are not independent. The interpretation of chi-square test:

I. For theorical or excel sheet purpose: If chi-square statistics is greater than critical value then reject the null hypothesis saying that 2 variables are dependent and if it's less, then accept the null hypothesis saying that 2 variables are independent.

II. While programming: If p-value is less than 0.05 then we reject the null hypothesis saying that 2 variables are dependent and if p-value is greater than 0.05 then we accept the null hypothesis saying that 2 variables are independent.

Here we did the test between categorical independent variables pairwise.

• If p-value<0.05 then remove the variable,

• If p-value>0.05 then keep the variable

variables which are highly dependent on each other based on p-values are:

- season and weathersit
- season and month
- holiday and weekday
- hoilday and workingday
- weekday and holiday
- weekday and workingday
- workingday and holiday
- workingday and weekday
- weathersit and season
- weathersit and mnth
- mnth and season
- mnth and weathersit

After analysing p value of all categorical features, we come to a conclusion that,
Variables which we have removed and kept:

Removed: mnth, weekday, workingday, weathersit.

Kept: season, holiday, yr

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

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

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

Hypothesis testing:

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

- If p-value is less than 0.05 then we reject the null hypothesis.
- And if p-value is greater than 0.05 then we accept the null hypothesis.

```
#anova test

anova_season =(lm(cnt ~ season, data = df))
summary(anova_season)
=Multiple R-squared:  0.347,    Adjusted R-squared:  0.3443
F-statistic: 128.8 on 3 and 727 DF,  p-value: < 2.2e-16


anova_year =(lm(cnt ~ yr, data = df))
summary(anova_year)
=Multiple R-squared:  0.3212,   Adjusted R-squared:  0.3202
F-statistic: 344.9 on 1 and 729 DF,  p-value: < 2.2e-16


anova_month =(lm(cnt ~ mnth, data = df))
summary(anova_month)
=Multiple R-squared:  0.3906,   Adjusted R-squared:  0.3813
F-statistic:  41.9 on 11 and 719 DF,  p-value: < 2.2e-16


anova_holiday =(lm(cnt ~ holiday, data = df))
summary(anova_holiday)
=Multiple R-squared:  0.004671, Adjusted R-squared:  0.003306
F-statistic: 3.421 on 1 and 729 DF,  p-value: 0.06476


anova_weekday =(lm(cnt ~ weekday, data = df))
summary(anova_weekday)
=Multiple R-squared:  0.006446, Adjusted R-squared:  -0.001788
F-statistic: 0.7829 on 6 and 724 DF,  p-value: 0.5835


anova_workingday =(lm(cnt ~ workingday, data = df))
summary(anova_workingday)
=Multiple R-squared:  0.00374,  Adjusted R-squared:  0.002373
F-statistic: 2.737 on 1 and 729 DF,  p-value: 0.09849


anova_weathersit =(lm(cnt ~ weathersit, data = df))
summary(anova_weathersit)
=Multiple R-squared:  0.09916,  Adjusted R-squared:  0.09668
F-statistic: 40.07 on 2 and 728 DF,  p-value: < 2.2e-16

anova_dteday =(lm(cnt ~ dteday, data = df))
```

summary(anova_dteday)
=Multiple R-squared: 0.009016, Adjusted R-squared: -0.03345
F-statistic: 0.2123 on 30 and 700 DF, p-value: 1

DELETE HOLIDAY, WEEKDAY,WORKINGDAY,DTEDAY
df = subset(df, select=-c(holiday,weekday,workingday,dteday))

2.1.4 Feature Scaling

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

- **Normalization**: Normalization refer to the dividing of a vector by its length. normalization normalizes the data in the range of 0 to 1. It is generally used when we are planning to use distance method for our model development purpose such as KNN. Normalizing the data improves convergence of such algorithms. Normalisation of data scales the data to a very small interval, where outliers can be loosed.
- **Standardization**: Standardization refers to the subtraction of mean from individual point and then dividing by its SD. Z is negative when the raw score is below the mean and Z is positive when above mean. When the data is distributed normally you should go for standardization.

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

Also, our most of our data is not distributed normally so we had choose normalization over standardization.

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

Note: It is performed only on Continuous variables.

Chapter 3

Modeling

The dependent variable can fall in any of the four categories:
1. Nominal
2. Ordinal
3. Interval
4. Ratio

If the dependent variable, XYZ, is nominal the only predictive analysis that we can perform is **Classification**, and if the dependent variable in our case (cnt) is Interval or Ratio the normal method is to do a **Regression** analysis.
You always start your model building from the simplest to more complex.

Splitting train and test Dataset

a. With the time series data, we will break up our train and test into continuous chunks.
b. The training data should be the earliest data and test data should be the latest data.
c. we will fit our model on the training data and test on the newest data, to understand how our model performs on new, unseen data.
d. we can't use sklearn's train_test_split bcoz it randomly shuffles the train and test data.

We have divided our data in 80-20% i.e. 80% in train and 20% in test.

```
#sampling
set.seed(12345)
t_index = sample(1:nrow(df), 0.8*nrow(df))
train = df[t_index,]
test = df[-t_index,]
```

LINEAR REGRESSION:-

```
#predictions on Train data set
LR_predict_train = predict(lr_model, dum_train_df[,-68])
plot(dum_train_df$cnt, LR_predict_train,
    xlab = 'Actual values',
    ylab = 'Predicted values',
```

```
    main = 'LR model')

#evaluation
postResample(LR_predict_train, dum_train_df$cnt)#R-sq = 0.85
mape(dum_train_df$cnt, LR_predict_train)
```

Test:-

```
#predictions on test
LR_predict_test = predict(lr_model, dum_test_df[,-68])
plot(dum_test_df$cnt, LR_predict_test,
    xlab = 'Actual values',
    ylab = 'Predicted values',
    main = 'LR model')

#evaluation
postResample(LR_predict_test, dum_test_df$cnt)#R-sq = 0.85
mape(dum_test_df$cnt, LR_predict_test)
postResample(LR_predict_test, dum_test_df$cnt)#R-sq = 0.85
        RMSE     Rsquared          MAE
7.416260e-12 1.000000e+00 6.925617e-12
> mape(dum_test_df$cnt, LR_predict_test)
[1] 2.121124e-13
```

DECISION TREE:=

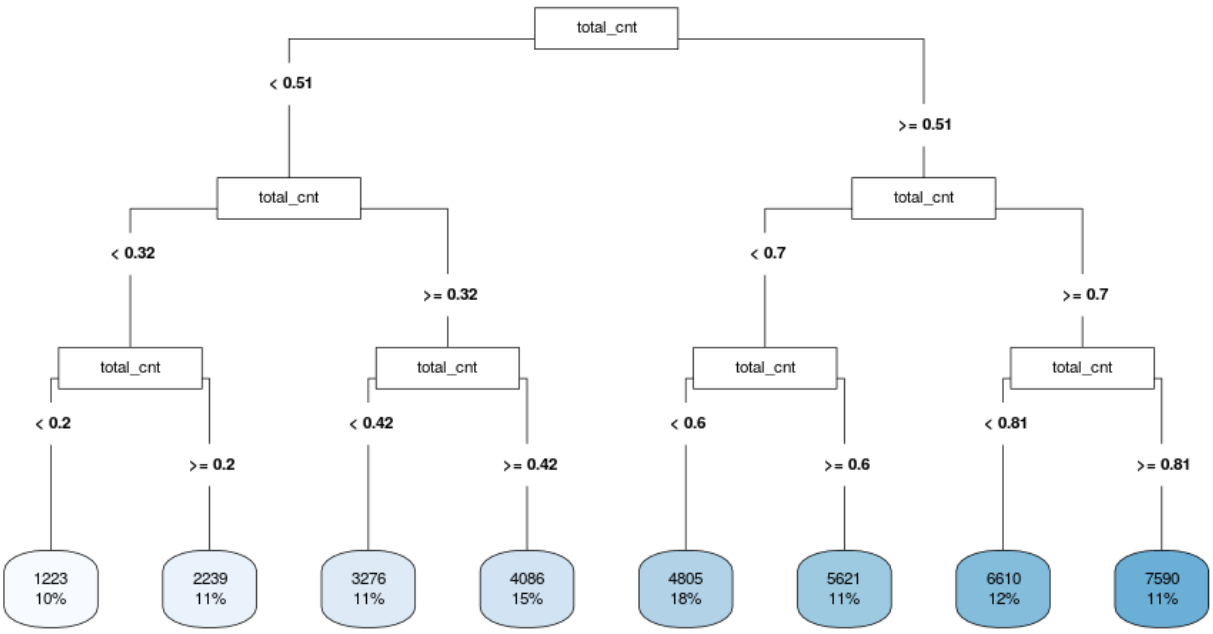


Fig :- Decision tree for cnt

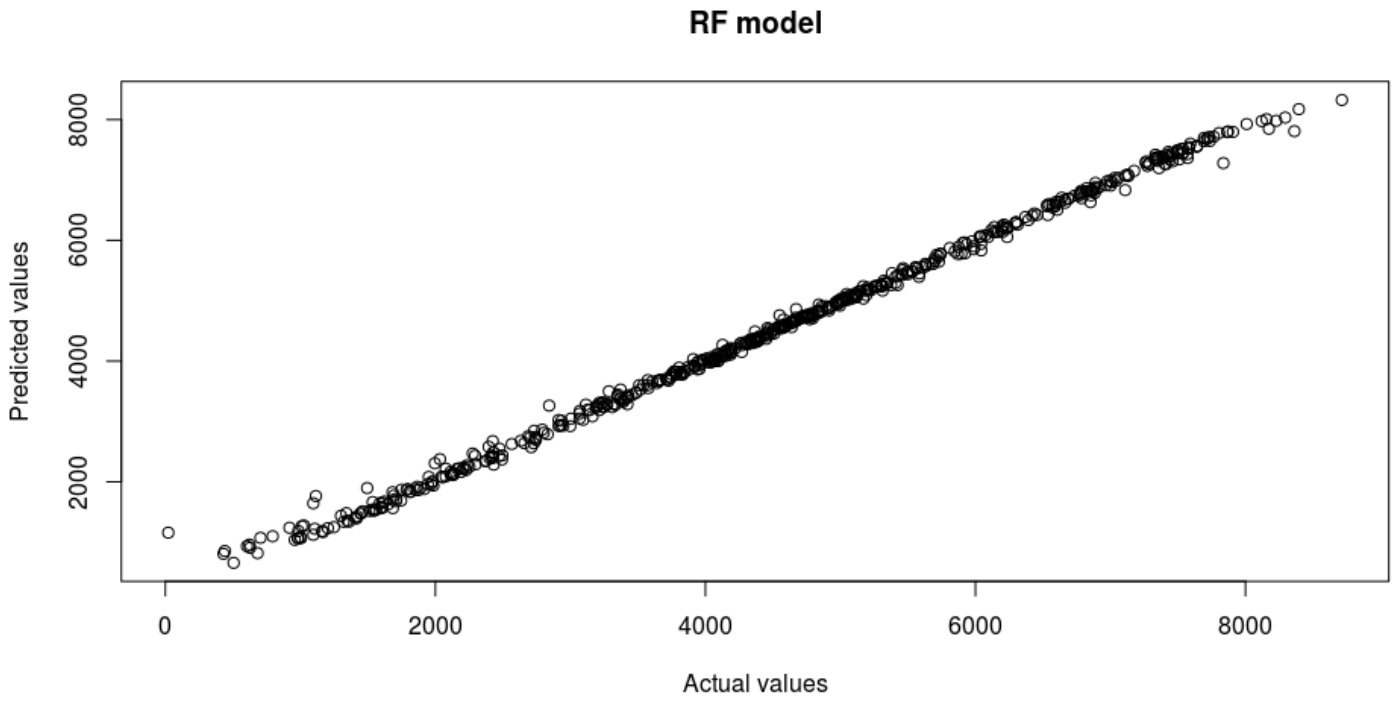
#evaluation
postResample(DT_Predict_train, train$cnt)
mape(train$cnt, DT_Predict_train)

```
postResample(DT_Predict_train, train$cnt)
       RMSE    Rsquared         MAE
276.9321662   0.9791681 227.3573836
> mape(train$cnt, DT_Predict_train)
[1] 17.18387
```

Train
#evaluation
postResample(DT_Predict_test, test$cnt)
mape(test$cnt, DT_Predict_test)

```
postResample(DT_Predict_test, test$cnt)
       RMSE    Rsquared         MAE
279.9179210   0.9804678 239.5357318
> mape(test$cnt, DT_Predict_test)
[1] 7.620063
```

RANDOM FOREST:-



#Train Result
postResample(RF_predict_train, train$cnt)#R-sq = 0.89
mape(train$cnt, RF_predict_train)

```
postResample(RF_predict_train, train$cnt)#R-sq = 0.89
       RMSE    Rsquared         MAE
109.4649404   0.9973985  60.5597234
> mape(train$cnt, RF_predict_train)
[1] 11.39759
```

#Test Result
postResample(RF_predict_test, test$cnt)#R-sq = 0.89
mape(test$cnt, RF_predict_test)

```
postResample(RF_predict_test, test$cnt)#R-sq = 0.89
       RMSE    Rsquared         MAE
189.0518521   0.9935447 134.9920050
> mape(test$cnt, RF_predict_test)
[1] 5.215196
```

Conclusion

We have selected Random Forest Regression as our Best Model to Predict Bike Rental Count.

```
# Final Model Random Forest


final_model = randomForest(cnt ~. , train, importance = TRUE, ntree = 500)
final_model

#error plotting
plot(final_model)

#Variable Importance plot
varImpPlot(final_model)

#Plotting predict train data using RF model
Final_predict_train = predict(final_model, train[,-9])
plot(train$cnt, Final_predict_train,
     xlab = 'Actual values',
     ylab = 'Predicted values',
     main = 'RF model')

#Train Result
postResample(Final_predict_train, train$cnt)#R-sq = 0.89
mape(train$cnt, Final_predict_train)



#Plotting predict test data using RF model
Final_predict_test = predict(final_model, test[,-9])
plot(test$cnt, Final_predict_test,
     xlab = 'Actual values',
postResample(Final_predict_train, train$cnt)#R-sq = 0.89
```

```
       RMSE    Rsquared         MAE
107.8014233   0.9974778  60.3059272
> mape(train$cnt, Final_predict_train)
[1] 10.69312
```

ylab = 'Predicted values',
main = 'RF model')

#Test Result
postResample(Final_predict_test, test$cnt)#R-sq = 0.89
mape(test$cnt, Final_predict_test)

```
postResample(Final_predict_test, test$cnt)#R-sq = 0.89
       RMSE    Rsquared         MAE
187.7548803   0.9934844 131.5158032
> mape(test$cnt, Final_predict_test)
[1] 5.099851
```

R Code:-

```
rm(list = ls())

setwd("H:\data scientist\Project1_Predicting_Bike_Rent")

# #loading Libraries
x = c("ggplot2", "corrgram", "DMwR", "usdm", "caret", "randomForest", "e1071",
      "DataCombine", "doSNOW", "inTrees", "rpart.plot", "rpart")

# #install.packages if not
lapply(x, install.packages)
#
#load Packages
lapply(x, require, character.only = TRUE)
rm(x)
######################################################################

#Loading Data
original_data = read.csv('day.csv',header = T,na.strings = c("","  ","NA"))
df = original_data    #Creating backup of orginal data

#############################################################################
#                EXPLORING DATA
#############################################################################

#viewing the data
head(df,4)
dim(df) #shape of data = row 731  ===  col = 16

#structure of data or data types
str(df)

#Summary of the data
summary(df)

#Carrying out date number
df$dteday <- format(as.Date(df$dteday,format="%Y-%m-%d"), "%d")

#removing instant
df$instant <- NULL

#unique value of each count
apply(df, 2,function(x) length(table(x)))

#Distribution of cnt variable
hist(df$cnt)

#Converting to proper dtype
```

```
cat_var = c('dteday','season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday','weathersit')
num_var = c('temp', 'atemp', 'hum', 'windspeed','casual','registered')

#Data Type Conversion Function
typ_conv = function(df,var,type){
  df[var] = lapply(df[var], type)
  return(df)
}

df = typ_conv(df,cat_var, factor)

#############################################################################
#         Checking Missing data
#############################################################################
apply(df, 2, function(x) {sum(is.na(x))}) #2 for columns as in R 1 = Row & 2 = Col

#Hence no missing data found

#############################################################################
#         Visualizing the data
#############################################################################
hist(df$casual)
hist(df$registered)
hist(df$cnt)

#library(ggplot2)
# CNT according to Season
ggplot(df, aes(fill=cnt, x=season)) +
  geom_bar(position="dodge") + labs(title="cnt ~ season")

# CNT according to holiday
ggplot(df, aes(fill=cnt, x=holiday)) +
  geom_bar(position="dodge") + labs(title="cnt ~ holiday")

# CNT according to season by yr
ggplot(df, aes(fill=cnt, x=season)) +
  geom_bar(position="dodge") + facet_wrap(~yr)+
  labs(title="CNT according to season by yr")

# CNT according to season by workingday
ggplot(df, aes(fill=cnt, x=season)) +
  geom_bar(position="dodge") + facet_wrap(~workingday)+
  labs(title="CNT according to season by workingday")

# CNT according to season by workingday
ggplot(df, aes(fill=cnt, x=workingday)) +
  geom_bar(position="dodge") + facet_wrap(~weekday)+
  labs(title="CNT according to workingday by weekday")

#################################################################
#             Outlier Analysis
#################################################################

#create Box-Plot for outlier analysis-
library(ggplot2)    #Library for visualization-
for(i in 1:length(num_var)){
  assign(paste0("AB",i),ggplot(aes_string(x="cnt",y=(num_var[i])),d=df)+
         geom_boxplot(outlier.color = "Red",outlier.shape = 18,outlier.size = 2,
```

```
                fill="Purple")+theme_get()+
        stat_boxplot(geom = "errorbar",width=0.5)+
        labs(x="Count of Bike",y=num_var[i])+
        ggtitle("Boxplot of count of bikes with",num_var[i]))
}

gridExtra::grid.arrange(AB1,AB2,AB3,ncol=3)
gridExtra::grid.arrange(AB4,AB5,ncol=2)

#Removing outlier by replacing with NA and then impute
for(i in c('hum', 'windspeed','casual')){
  print(i)
  outv = df[,i][df[,i] %in% boxplot.stats(df[,i])$out]
  print(length(outv))
  df[,i][df[,i] %in% outv] = NA
}

#checking all the missing values
#library(DMwR)
sum(is.na(df))

df$hum[is.na(df$hum)] = mean(df$hum,na.rm = T)

df$casual[is.na(df$casual)] = mean(df$casual,na.rm = T)

df$windspeed[is.na(df$windspeed)] = mean(df$windspeed, na.rm = T)

#df = knnImputation(df, k=3)

sum(is.na(df))
######################################################################
#             Feature Selection
######################################################################

#Here we will use corrgram library to find corelation

##Correlation plot
library(corrgram)
num_var = c('temp', 'atemp', 'hum', 'windspeed','casual','registered','cnt')

corrgram(df[,num_var],
        order = F,  #we don't want to reorder
        upper.panel=panel.pie,
        lower.panel=panel.shade,
        text.panel=panel.txt,
        main = 'CORRELATION PLOT')
#We can see var the highly corr related var in plot marked dark blue.
#Dark blue color means highly positive cor related

df = subset(df, select=-c(atemp,casual,registered))

# #Checking dependency among different categorical variables
cat_var = c('dteday','season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday','weathersit')
cat_df = df[,cat_var]

for (i in cat_var){
  for (j in cat_var){
    print(i)
```

```
   print(j)
   print(chisq.test(table(cat_df[,i], cat_df[,j]))$p.value)
  }
}

#anova test

anova_season =(lm(cnt ~ season, data = df))
summary(anova_season)

anova_year =(lm(cnt ~ yr, data = df))
summary(anova_year)

anova_month =(lm(cnt ~ mnth, data = df))
summary(anova_month)

anova_holiday =(lm(cnt ~ holiday, data = df))
summary(anova_holiday)

anova_weekday =(lm(cnt ~ weekday, data = df))
summary(anova_weekday)

anova_workingday =(lm(cnt ~ workingday, data = df))
summary(anova_workingday)

anova_weathersit =(lm(cnt ~ weathersit, data = df))
summary(anova_weathersit)

anova_dteday =(lm(cnt ~ dteday, data = df))
summary(anova_dteday)


################################################
# #check multicollearity
################################################
#Linear Regression
library(usdm)

vif(df)

df = subset(df, select=-c(holiday,weekday,workingday,dteday))
#dteday

#####################################
## Feature Scaling
#####################################
#min(df$cnt) ----->  22
#max(df$cnt) ----->  8714
hist(df$cnt)
colnames(df)

# #Normalization of cnt
df$total_cnt = (df$cnt - min(df$cnt)) / (max(df$cnt) - min(df$cnt))

#####################################################################
#             Sampling of Data
#####################################################################
```

```
#sampling
set.seed(12345)
t_index = sample(1:nrow(df), 0.8*nrow(df))
train = df[t_index,]
test = df[-t_index,]



#Removing All the custom variable from memory
#library(DataCombine)
rmExcept(c("test","train","original_data",'df'))

library(caret)

#mape
mape = function(actual, predict){
  mean(abs((actual-predict)/actual))*100
}

##########################################
# # # ??? Linear Regression
# ######################################
#
##Linear regression
dumy = dummyVars(~., df)
dummy_df = data.frame(predict(dumy, df))

set.seed(101)
dum_index = sample(1:nrow(dummy_df), 0.8*nrow(dummy_df))
dum_train_df = dummy_df[dum_index,]
dum_test_df = dummy_df[-dum_index,]

#Linear model
lr_model = lm(cnt ~. , data = dum_train_df)
summary(lr_model)

#predictions on Train data set
LR_predict_train = predict(lr_model, dum_train_df[,-68])
plot(dum_train_df$cnt, LR_predict_train,
    xlab = 'Actual values',
    ylab = 'Predicted values',
    main = 'LR model')

#evaluation
postResample(LR_predict_train, dum_train_df$cnt)#R-sq = 0.85
mape(dum_train_df$cnt, LR_predict_train)



#predictions on test
LR_predict_test = predict(lr_model, dum_test_df[,-68])
plot(dum_test_df$cnt, LR_predict_test,
    xlab = 'Actual values',
    ylab = 'Predicted values',
    main = 'LR model')

#evaluation
postResample(LR_predict_test, dum_test_df$cnt)#R-sq = 0.85
mape(dum_test_df$cnt, LR_predict_test)
```

```
# ########################################
# # # # Decision Tree
# # ########################################
#
##Decison tree

library(rpart.plot)
library(rpart)

set.seed(121)
#model
dt_model = rpart(cnt~. , data = train, method = "anova")
summary(dt_model)
plt = rpart.plot(dt_model, type = 5, digits = 2, fallen.leaves = TRUE)

#predictions on train
DT_Predict_train = predict(dt_model, train[,-9])
plot(train$cnt, DT_Predict_train,
    xlab = 'Actual values',
    ylab = 'Predicted values',
    main = 'DT model')

#evaluation
postResample(DT_Predict_train, train$cnt)
mape(train$cnt, DT_Predict_train)

#predictions on test
DT_Predict_test = predict(dt_model, test[,-9])
plot(test$cnt, DT_Predict_test,
    xlab = 'Actual values',
    ylab = 'Predicted values',
    main = 'DT model')

#evaluation
postResample(DT_Predict_test, test$cnt)
mape(test$cnt, DT_Predict_test)


# ########################################
# # # # ??? Random Forest
# # ########################################
#
##Random forest
library(randomForest)
library(inTrees)
set.seed(101)
#model
rf_model = randomForest(cnt ~. , train, importance = TRUE, ntree = 500)
rf_model

#error plotting
plot(rf_model)

#Variable Importance plot
varImpPlot(rf_model)
```

```
#Plotting predict train data using RF model
RF_predict_train = predict(rf_model, train[,-9])
plot(train$cnt, RF_predict_train,
    xlab = 'Actual values',
    ylab = 'Predicted values',
    main = 'RF model')

#Train Result
postResample(RF_predict_train, train$cnt)#R-sq = 0.89
mape(train$cnt, RF_predict_train)


#Plotting predict test data using RF model
RF_predict_test = predict(rf_model, test[,-9])
plot(test$cnt, RF_predict_test,
    xlab = 'Actual values',
    ylab = 'Predicted values',
    main = 'RF model')

#Test Result
postResample(RF_predict_test, test$cnt)#R-sq = 0.89
mape(test$cnt, RF_predict_test)



#K-fold cross validation function
kfold_train <- function(model){
  x=trainControl(method = "cv",number = 10)
  model= train(cnt ~.,data=train,metric="RMSE",method=model,trControl=x)
  print(model)
  return(model)
}
result <- function(model){
  model = model
  set.seed(101)
  pred = predict(model, train[,-9])
  print(postResample(pred, train$cnt))
  print(mape(train$cnt, pred))

  print('Test Results____')
  pred = predict(model,test[,-9])
  print(postResample(pred, test$cnt))
  print(mape(test$cnt, pred))
}

# ############################################################
# ## Final Model Random Forest
# ############################################################
#
final_model = randomForest(cnt ~. , train, importance = TRUE, ntree = 500)
final_model

#error plotting
plot(final_model)

#Variable Importance plot
varImpPlot(final_model)
```

```
#Plotting predict train data using RF model
Final_predict_train = predict(final_model, train[,-9])
plot(train$cnt, Final_predict_train,
    xlab = 'Actual values',
    ylab = 'Predicted values',
    main = 'RF model')

#Train Result
postResample(Final_predict_train, train$cnt)#R-sq = 0.89
mape(train$cnt, Final_predict_train)


#Plotting predict test data using RF model
Final_predict_test = predict(final_model, test[,-9])
plot(test$cnt, Final_predict_test,
    xlab = 'Actual values',
    ylab = 'Predicted values',
    main = 'RF model')

#Test Result
postResample(Final_predict_test, test$cnt)#R-sq = 0.89
mape(test$cnt, Final_predict_test)

#############################################################################
## Saving the output
rmExcept(c("final_model",'mape',"original_data",'df'))
df$predict_cnt <-  round(predict(final_model, df[,-9]))
original_data$predict_cnt <- df$predict_cnt
write.csv(original_data, 'output_R.csv',row.names = F)
write.csv(original_data[,c("dteday","weathersit","season","mnth",'temp',"hum","windspeed","cnt","predict_cnt
")], 'output_R.csv',row.names =
```