BIKE RENTAL COUNT PREDICTION

HIMANSHU GUPTA

12-01-2020

Meerut Institute of Engineering & Technology, Meerut

Table of Contents

Chapter 1

Introduction

Whether it's to boost your fitness, health or bank balance, or as an environmental choice, taking up bicycle riding could be one of the best decisions you ever make. Remember the days of the bicycle built for two, when tourists rented bikes to explore island areas where cars either didn't exist or were blessedly limited? Those days are still here, but the majority of bicycle rental businesses are clustered around heavily trafficked tourist spots.

However, with increased rails-to-trails projects and traffic congestion there are many more bicycle paths away from resort areas, office space, residential area that are creating excellent new rental opportunities. Many bicycle rental shops are now featuring inline skate rentals as well, especially in places like USA, UK. The bike rental service has a great potential as a business opportunity.

1.1 Problem Statement

The objective of this case study is the prediction of bike rental count on daily based on the environmental and seasonal settings. The dataset contains 731 observations, 15 predictors and 1 target variable. The predictors are describing various environment factors and settings like season, humidity etc. We need to build a prediction model to predict estimated count or demand of bikes on a particular day based on the environmental factors.

1.2 Dataset

The data set consist of 731 observation recorded over a period of 2 years, between 2011 and 2012. It has 15 predictors or variables and 1 target variable. All the variables are described in table 1.

Variable names	Description
Instant	Record index
Dteday	Date
Season	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 from Holiday Schedule)
Weekday	Day of the week
Workingday	If day is neither weekend nor holiday is 1, otherwise is 0.
weathersit	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 + Scattered clouds 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_maxt- 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

Table1. Description of variables

The data set consist of 7 continuous and 8 categorical variables. Sample data is shown below.

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit
1	1/1/2011	1	0	1	0	6	0	2
2	1/2/2011	1	0	1	0	0	0	2
3	1/3/2011	1	0	1	0	1	1	1
4	1/4/2011	1	0	1	0	2	1	1
5	1/5/2011	1	0	1	0	3	1	1
6	1/6/2011	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

Table2. Sample data

Chapter 2

Methodology

The solution of this problem is divided into three parts. First was EDA (Exploratory Data analysis) and pre-processing, followed by modelling and performance tuning and comparison. During first part data pre-processing step like missing value analysis, outlier analysis, univariate and bi-variate analysis etc. were performed. After that data was split into train and test. The target variable is a continuous variable, so it a regression problem. Linear regression and Random forest regression were used for modelling and their performance comparison was performed. Both the algorithms were implemented in R and python.

2.1 Pre-Processing and EDA

Pre-processing and EDA was performed in both R and python. The dataset consists of 731 observations, and 15 predictors. The process of pre-processing and EDA is described below.

2.1.1 Target Variable – 'cnt'

The target variable in the problem statement is the total count of registered and casual users of bikes on a single day. '*cnt'* is the combined value of *'registered'* and *'casual'* variables. The histogram, distribution and summary statistics of *'cnt'* are as follow.

Summary Stats	Values
count	731
mean	4504.34
std	1937
min	22
25%	3152
50%	4548
75%	5956
max	8714

Table3. Summary statistics of target variable '*cnt'*

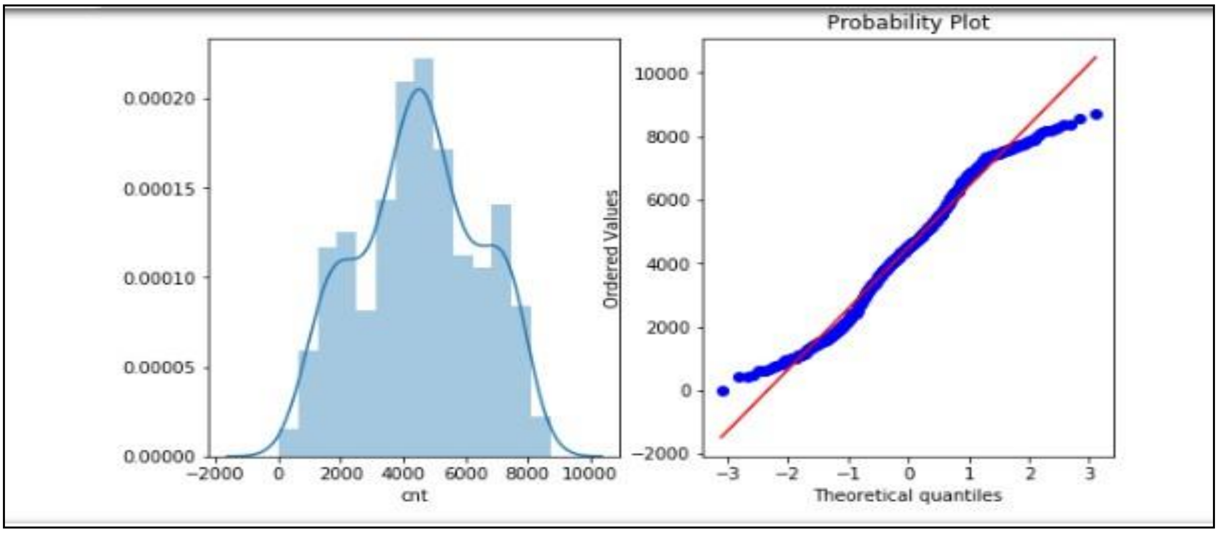


Figure1. Target variable distribution

2.1.2 Missing value Analysis

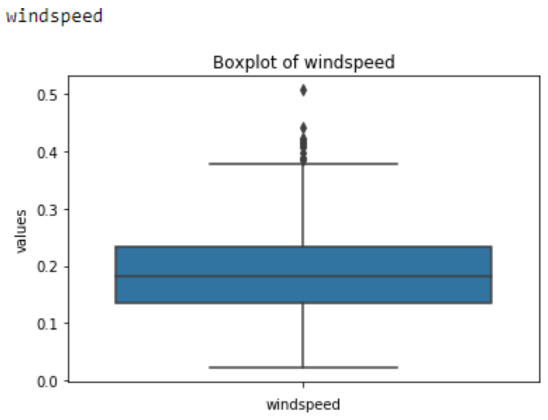
Missing values are the data which is not present in the particular variable or observations. It may happen due to human error, or it may mark as an optional during the survey. If the data set contains missing values which is above 30%, either we need to drop the column or that particular observation.in our dataset we don't have any missing values but in real world problems there is always some missing values.no missing values were found for this dataset.

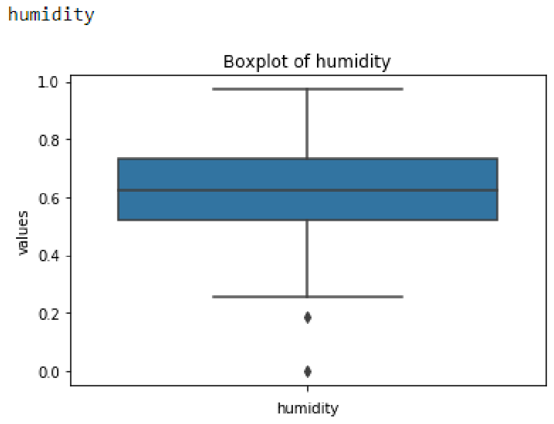
```
season         0
year           0
month          0
holiday        0
weekday        0
workingday     0
weather        0
temperature    0
atemp          0
humidity       0
windspeed      0
count          0
dtype: int64
```

2.1.3 Outlier Analysis:

Basically outliers are the values which are lying far away from the remaining variables which may lead biased towards the higher value which results in the performance of our model. So that we need to treat the outliers .

Here outliers are detected using boxplot. We have inliers in humidity and outliers in windspeed other than that we don't have any outliers.so, In our case we saved minimum value to the inliers and maximum values to the outliers.so that we no need to loss the data and also we can increase the performance the of our model. How much data we feed is that much accuracy to our model.



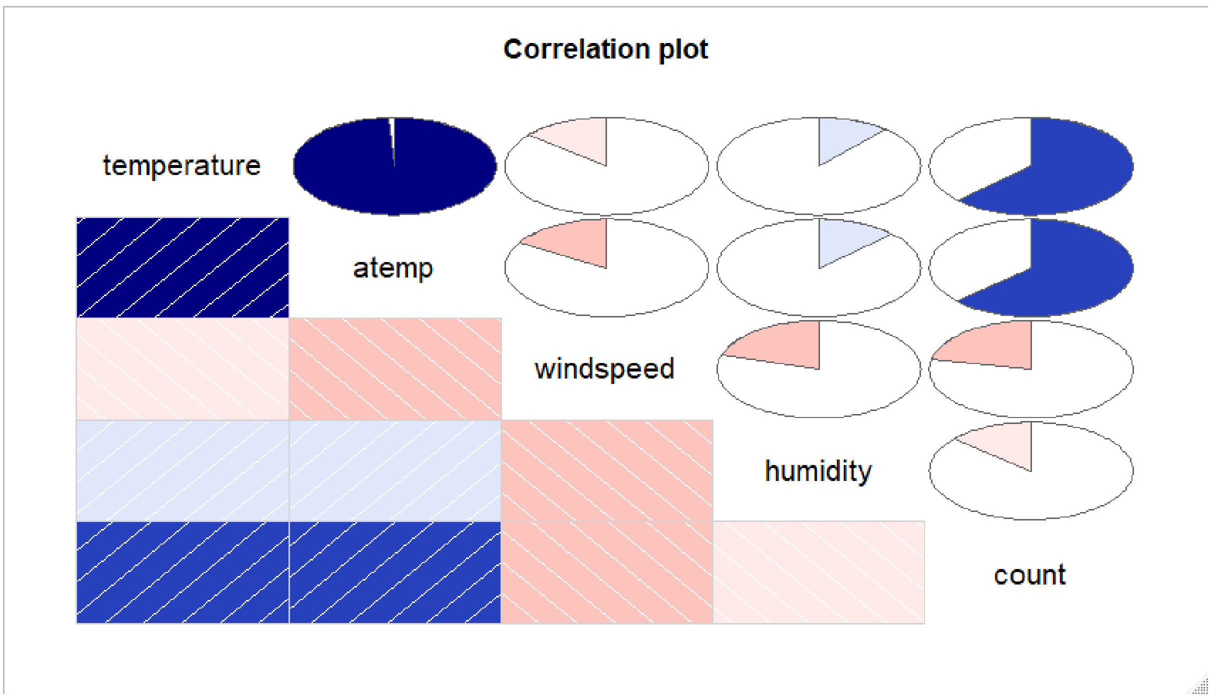


2.1.4_Feature Selection:

We can use correlation analysis for numerical variables and Analysis of Variance for categorical variables. It shows correlation between the two variables. So that if two variables carrying same information can be removed.

2.1.4a: Correlation matrix and plot

```
            temperature      atemp  windspeed
temperature   1.0000000  0.9917378 -0.1401690
atemp         0.9917378  1.0000000 -0.1660383
windspeed    -0.1401690 -0.1660383  1.0000000
humidity      0.1141910  0.1265874 -0.2044964
count         0.6258917  0.6292045 -0.2161933
              humidity      count
temperature  0.1141910  0.6258917
atemp        0.1265874  0.6292045
windspeed   -0.2044964 -0.2161933
humidity     1.0000000 -0.1366214
count       -0.1366214  1.0000000
> #correlation plot
```



From the above plot, we say that temperature and atemp variables are carrying same information. So we need to remove atemp variable.

2.1.4b: ANALYSIS OF VARIANCE:

```
                 sum_sq     df           F         PR(>F)
season    4.517974e+08    1.0  143.967653   2.133997e-30
Residual  2.287738e+09  729.0         NaN            NaN
                 sum_sq     df           F         PR(>F)
year      8.798289e+08    1.0  344.890586   2.483540e-63
Residual  1.859706e+09  729.0         NaN            NaN
                 sum_sq     df           F         PR(>F)
month     2.147445e+08    1.0  62.004625   1.243112e-14
Residual  2.524791e+09  729.0        NaN            NaN
                 sum_sq     df          F    PR(>F)
holiday   1.279749e+07    1.0  3.421441  0.064759
Residual  2.726738e+09  729.0       NaN       NaN
                 sum_sq     df          F    PR(>F)
weekday   1.246109e+07    1.0  3.331091  0.068391
Residual  2.727074e+09  729.0       NaN       NaN
                   sum_sq     df          F    PR(>F)
workingday  1.024604e+07    1.0  2.736742  0.098495
Residual    2.729289e+09  729.0       NaN       NaN
                 sum_sq     df          F         PR(>F)
weather   2.422888e+08    1.0  70.729298   2.150976e-16
Residual  2.497247e+09  729.0        NaN            NaN
```

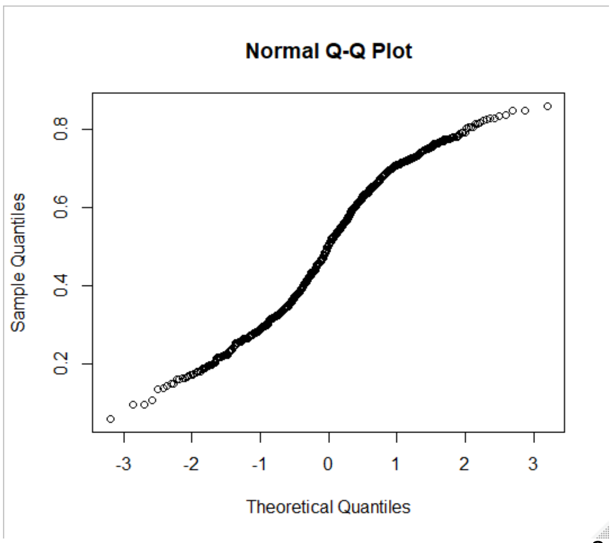
From the above diagram, holiday,weekday,and working day these variables has p-value which is higher than 0.05. so that we need to drop these variables.

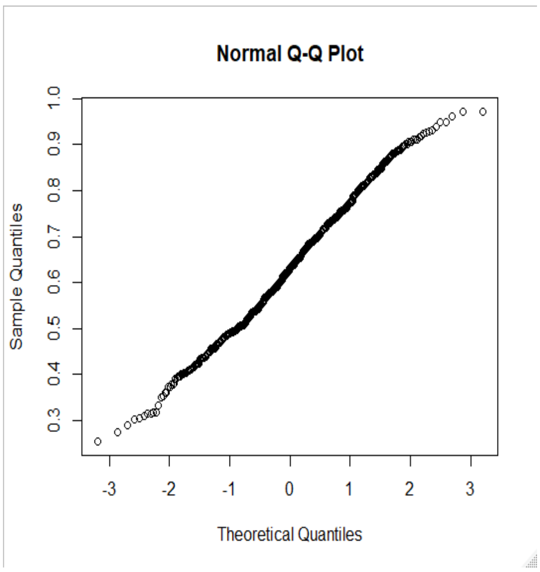
After the dimension reduction we have only 8 variables:

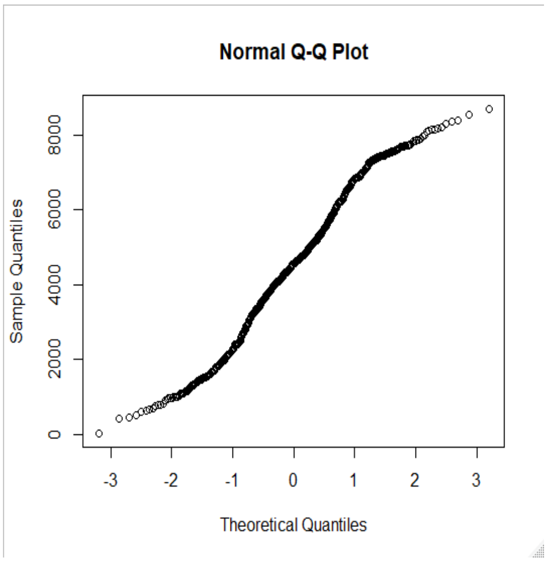
Temp,hum,windspeed,cnt,season,yr,mnth,weathersit

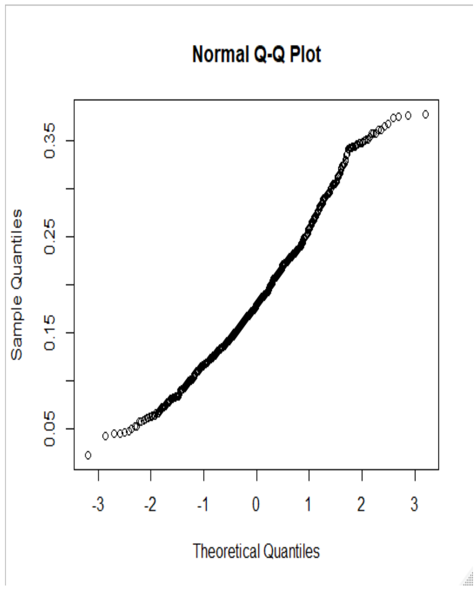
2.1.5_ Feature Scaling:

In our dataset, all our continuous variables are already normalized. So we don't need to need any scaling methods to scale the data. Though we can use qqplot, summary, distribution of the data .









2.1.6_Bivariate Analysis

In bivariate analysis, we will look at the relationship between target variable and predictor. First we look for continuous variables.

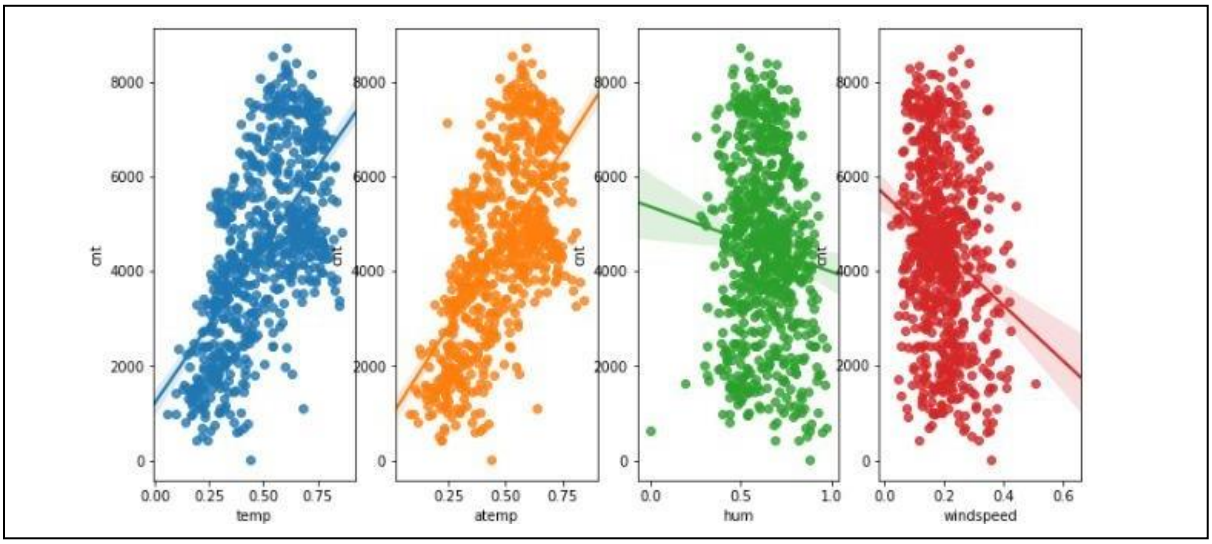


Figure 10. relationship between target variable and continuous predictors

From the above scatter plots, we can see that

A) 'cnt' and 'temp' have strong and positive relationship. It means that as the temperature rises, the bike demand also increase.

B) 'atemp' and 'cnt' have strong and positive relationship. It means that as the ambient temperature rise, demand for bikes also increases.

C) 'hum' (humidity) has a negative linear relationship with 'cnt'. As humidity increases, count decreases.

D) 'windspeed' has negative linear relationship with 'cnt'. With an increase in windspeed, bike count decreases.

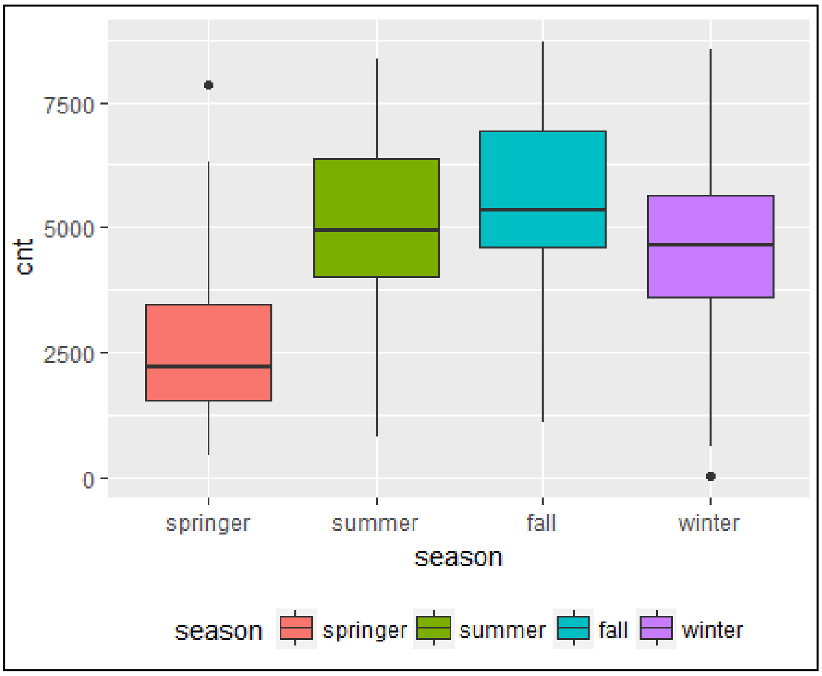


Figure 2 . relationship between season and count

Figure 2 is showing relationship between count (demand) and season.

1.The count is highest for fall season and lowest for spring season.
2.There is no significance difference between count for summer and fall.

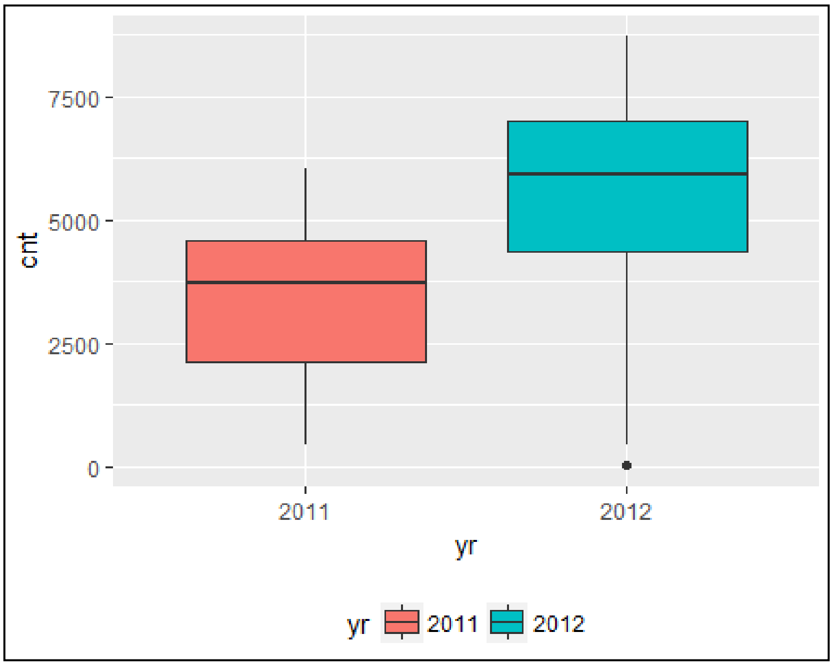
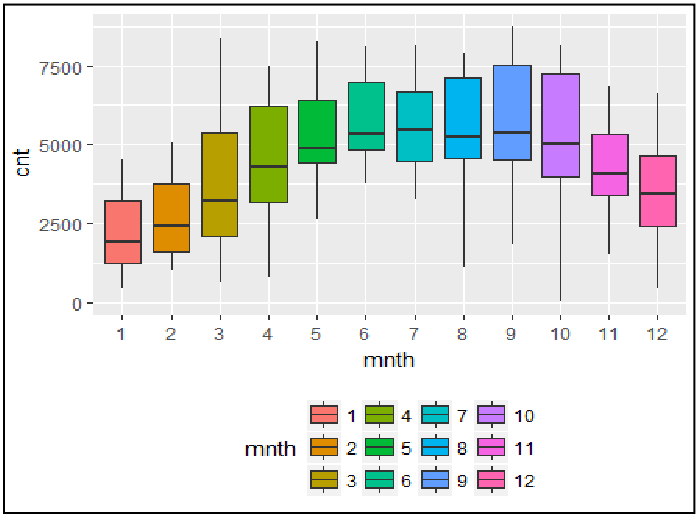


Figure 3. relationship between year and count

Figure 3 is showing that bike demand was higher in 2012 as compared with 2011.



- From figure 4 it can be inferred that count is high in the month of august, September and October.
- lowest count is for January ad February.
- We can see that as the weather changes from cold to hot, count also

Figure 4. relationship between months and count

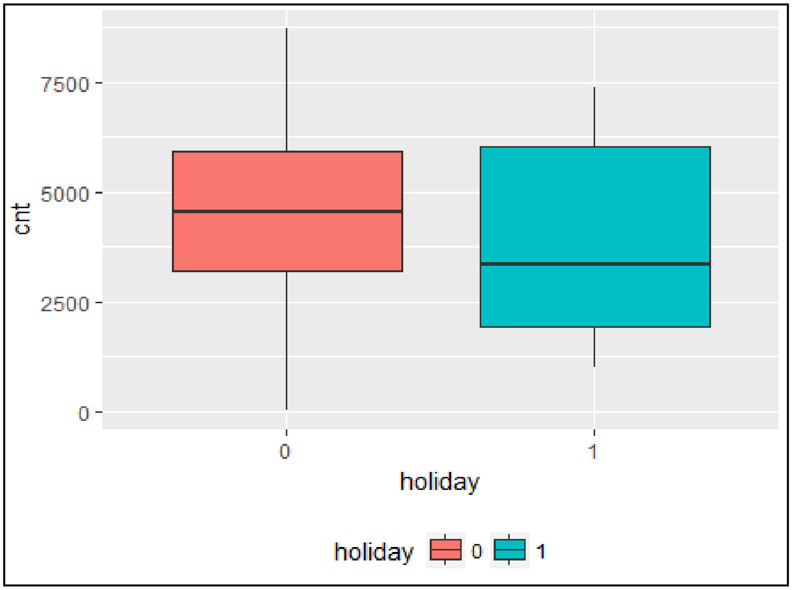
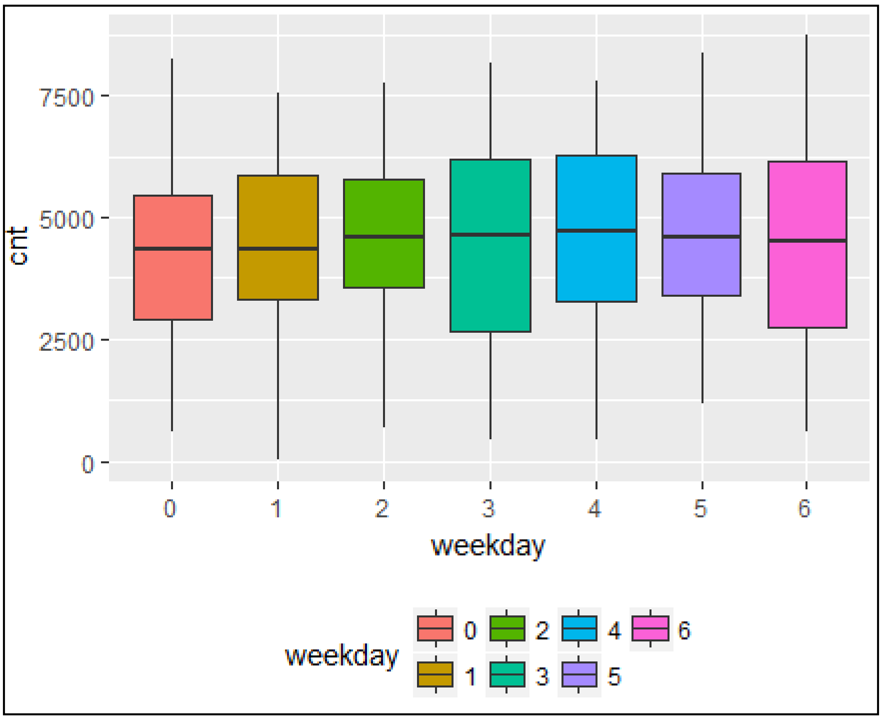


Figure 5. relationship between holidays and count

From the boxplot it is visible that count and it’s median is higher on holidays. People prefer to rent bike on holiday.



There is not much variation in median of count on weekdays. They are nearly similar on all weekdays.

Figure 6. relationship between weekdays and count

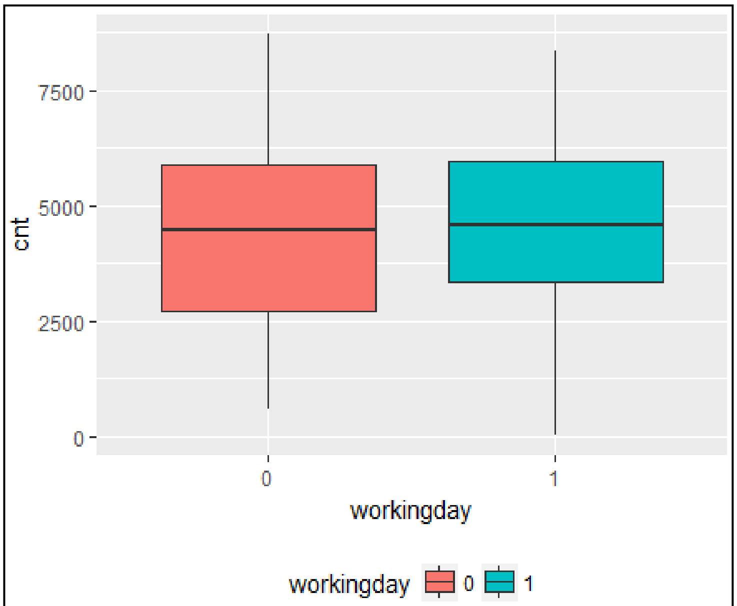


Figure 7. relationship between workingday and count

1.There is median for count is same for working and non-working days.

2.The range is longer for non- working days.

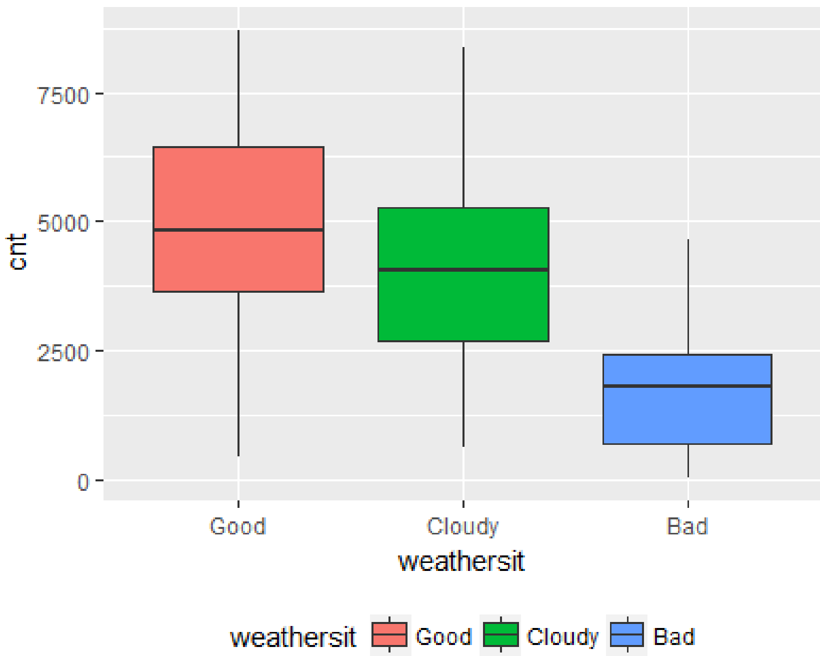


Figure 8: relationship between weathersit and count

1. The count is maximum when weather situation is good.
2. It is least when weather conditions are bad.

2.2 Modeling:

Next we need to split the data into train and test data and build a model using train data to predict the output using test data. Different models to be built and the model which gives more accurate values must be selected.

2.1.1 LINEAR REGRESSION:

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things:

(1) Does a set of predictor variables do a good job in predicting an outcome (dependent) variable?
(2) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates– impact the outcome variable?

These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. We trained our model in both R and Python and predicted in these languages using test data.

2.1.2 DECISION TREE:

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

2.1.3 RANDOM FOREST:

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees, which involves training each decision tree on a different data sample where sampling is done with replacement. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.The higher no of trees in the random forest will give higher no of accuracy, so in random forest we can go for multiple trees. It can handle large no of independent variables without variable deletion and it will give the estimates that what variables are important.

3. Model Evaluation:

3.1. EVALUATION METRICS:

In regression problems, we have three important metrics.they are

MAPE(Mean Absolute Percentage Error

R-SQUARED

RMSE(Root Mean Square Error)

3.1.1 MAPE(Mean Absolute Percentage Error)

MAPE is a measure of prediction accuracy of a forecasting method. It measures accuracy in terms of percentage.Lower value of MAPE indicates better fit.

3.1.2 R-SQUARED

R-squared is basically explains the degree to which input variable explain the variation of the output. In simple words Rsquared tells how much variance of dependent variable explained by the independent variable. It is a measure if goodness of fit in regression line. Higher values of R-square indicate better fit.

3.1.3 RMSE(Root Mean Square Error)

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors).Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit.

3.2 MODEL SELECTION:

From the predicted output in R and Python, the random forest model can have explained almost 90% of the predictor matches with the target variable.the values of the random forest model is mentioned below.

- **MAPE = 0.12**
- **R-SQUARED =0.91**
- **RMSE = 593.84**

R code:

```
# bike rental count prediction

#first clean R enviorment
rm(list=ls(all=T))

#set  working directory
setwd("C:/Users/himanshu gupta/Desktop/edwisor/project/2")
getwd()

#Load Libraries
```

```
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies",
"e1071", "Information",

      "MASS", "rpart", "gbm", "ROSE", "sampling", "DataCombine",
"inTrees","gridExtra","scales","psych","gplots")


#install.packages(x)

lapply(x, require, character.only = TRUE)

rm(x)


#lets load the data

bike_rental_data = read.csv("day.csv")


#-----------------------------------------------------#

#               explore the data                  #


dim(bike_rental_data)

names(bike_rental_data)

head(bike_rental_data)

str(bike_rental_data)

summary(bike_rental_data)


#in our dataset some variables have no useful information for our prediction
```

```
#so it is better to remove those variables.it helps us to make useful inferences

#lets drop unnecessary variables
bike_rental_data = subset(bike_rental_data,select = -c(instant,dteday,casual,registered))

#------------------------------------------------------#
#               data-preprocessing                  #

#missing value analysis
sapply(bike_rental_data, function(x) {
  sum(is.na(x))
})
# there are no missing values

#outlier analysis

cnames=c("temp",'atemp','windspeed','hum','cnt')
for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"), data = subset(bike_rental_data))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="green", fill = "grey" ,outlier.shape=18,
```

```
                outlier.size=1, notch=FALSE) +
        theme(legend.position="bottom")+
        labs(y=cnames[i],x="count")+
        ggtitle(paste("Box plot of count for",cnames[i])))
}

#plotting boxplot
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,ncol=2)
gridExtra::grid.arrange(gn5,ncol=1)

#lets remove outliers using boxplot
df = bike_rental_data
for(i in cnames){
  print(i)
  outliers = bike_rental_data[,i][bike_rental_data[,i] %in% boxplot.stats(bike_rental_data[,i])$out]
  print(length(outliers))
  bike_rental_data = bike_rental_data[which(!bike_rental_data[,i] %in% outliers),]
}

#lets plot boxplot after removing outliers
for (i in 1:length(cnames))
```

```
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"), data = subset(bike_rental_data))+
         stat_boxplot(geom = "errorbar", width = 0.5) +
         geom_boxplot(outlier.colour="green", fill = "grey" ,outlier.shape=18,
                      outlier.size=1, notch=FALSE) +
         theme(legend.position="bottom")+
         labs(y=cnames[i],x="cnt")+
         ggtitle(paste("Box plot of cnt for",cnames[i])))
}
#plotting Boxplot after removing outliers
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,ncol=2)
gridExtra::grid.arrange(gn5,ncol=1)


#------------------------------------------------------#
#              feature selection                  #


#find correlation matrix using corrplot and correlation plot using corrgram library
#FOR NUMERICAL VARIABLES


#save dataset after outlier analysis
df = bike_rental_data
```

```
#correlation matrix

cnames=c("temp","atemp","windspeed","hum")

sapply(bike_rental_data,class)

correlation_matrix = cor(bike_rental_data,cnames)

correlation_matrix

#correlation plot

corrgram(bike_rental_data[,cnames],order = F,upper.panel = panel.pie,

        text.panel = panel.txt,main = 'Correlation plot')


#From the correlation plot,we see that temp and atemp variables are correlated to each other

#so we need to remove atemp variable.


#perform annova test for categorical variables

catnames=c('season','yr','mnth','holiday','weekday','workingday','weathersit')

for (i in catnames) {

  print(i)

  anova = summary(aov(formula = cnt~bike_rental_data[,i],bike_rental_data))

  print(anova)

}

#based on the anova result, we can drop three variables named,

# holiday, weekday, workingday

#because these variables having the p-value > 0.05
```

```
#Dimension reduction

bike_rental_data = subset(bike_rental_data,select = -c(holiday,weekday,workingday,atemp))


#lets check after dimension reduction

dim(bike_rental_data)

head(bike_rental_data)


#-----------------------------------------------------#

#               feature scaling                  #


#check normality between the varaibles


cnames=c("temp","windspeed","hum","cnt")

for (i in cnames){

  print(i)

  normality = qqnorm(bike_rental_data[,i])


}


#already we plotted distrution between these variables,lets recall it

for(i in 1:length(cnames))
```

```
{
  assign(paste0("h",i),ggplot(aes_string(x=(cnames[i])),
                              data=subset(bike_rental_data))+
           geom_histogram(fill="blue",colour = "green")+geom_density()+
           scale_y_continuous(breaks =pretty_breaks(n=8))+
           scale_x_continuous(breaks = pretty_breaks(n=8))+
           theme_bw()+xlab(cnames[i])+ylab("Frequency")+
           ggtitle(paste("distribution plot for ",cnames[i])))
}
gridExtra::grid.arrange(h1,h2,h3,h4,ncol = 2)

#summary of the data
for (i in cnames) {
  print(i)
  print(summary(bike_rental_data[,i]))

}
#Based on the above inferences and plots,we can see that the variables are normalised.

# bivariate analysis for categorical variables
bivariate_categorical <-
  function(dataset, variable, targetVariable) {
```

```
    variable <- enquo(variable)
    targetVariable <- enquo(targetVariable)

    ggplot(
      data = dataset,
      mapping = aes_(
        x = rlang::quo_expr(variable),
        y = rlang::quo_expr(targetVariable),
        fill = rlang::quo_expr(variable)
      )
    ) +
      geom_boxplot() +
      theme(legend.position = "bottom") -> p
    plot(p)

  }

bivariate_continous <-
  function(dataset, variable, targetVariable) {
    variable <- enquo(variable)
    targetVariable <- enquo(targetVariable)
    ggplot(data = dataset,
```

```
        mapping = aes_(
          x = rlang::quo_expr(variable),
          y = rlang::quo_expr(targetVariable)
        )) +
    geom_point() +
    geom_smooth() -> q
  plot(q)

}

bivariate_categorical(bike_rental_data, season, cnt)
bivariate_categorical(bike_rental_data, yr, cnt)
bivariate_categorical(bike_rental_data, mnth, cnt)
bivariate_categorical(bike_rental_data, weathersit, cnt)
bivariate_continous(bike_rental_data, temp, cnt)
bivariate_continous(bike_rental_data, hum, cnt)
bivariate_continous(bike_rental_data, windspeed, cnt)


#-----------------------------------------------------#
#               model devlopment                      #
```

```
#we can not pass categorical variables to regression problems

#so convert categorical variables into dummy variables

#saving our preprocessed data

df = bike_rental_data


#create dummies

library(dummies)

catnames = c('season','yr','mnth','weathersit')

bike_rental_data = dummy.data.frame(bike_rental_data,catnames)


#we have created dummies,lets check dimension and top 5 observations

dim(bike_rental_data)

head(bike_rental_data)


#divide the data into train and test

set.seed(1234)

train_index = sample(1:nrow(df), 0.8 * nrow(df))

train_data = bike_rental_data[train_index,]

test_data = bike_rental_data[-train_index,]


#----------------------------------------------------#
```

 #   (1) linear regression   #

```
#running regression model
lm_model = lm(cnt~. ,data = bike_rental_data)
#lets check performance of our modedl
summary(lm_model)
#Residual standard error: 787.3 on 696 degrees of freedom
#Multiple R-squared:  0.8388,	Adjusted R-squared:  0.8342
#F-statistic: 181.1 on 20 and 696 DF,  p-value: < 2.2e-16

# Function for Error metrics to calculate the performance of model
#lets build function for MAPE
#calculate MAPE
MAPE = function(y, y1){
  mean(abs((y - y1)/y))
}

# Function for r2 to calculate the goodness of fit of model
rsquare=function(y,y1){
  cor(y,y1)^2
}

# Function for RMSE value
```

```
RMSE = function(y,y1){
  difference = y - y1
  root_mean_square = sqrt(mean(difference^2))
}

#lets predict for train and test data
Predictions_LR_train = predict(lm_model,train_data[,-25])
Predictions_LR_test = predict(lm_model,test_data[,-25])

#let us check performance of our model

#mape calculation
LR_train_mape = MAPE(Predictions_LR_train,train_data[,25])
LR_test_mape = MAPE(test_data[,25],Predictions_LR_test)

#Rsquare calculation
LR_train_r2 = rsquare(train_data[,25],Predictions_LR_train)
LR_test_r2 = rsquare(test_data[,25],Predictions_LR_test)

#rmse calculation
LR_train_rmse = RMSE(train_data[,25],Predictions_LR_train)
LR_test_rmse = RMSE(test_data[,25],Predictions_LR_test)
```

```
print(LR_train_mape) #0.15

print(LR_test_mape) #0.18

print(LR_train_r2) #0.831

print(LR_test_r2) #0.867

print(LR_train_rmse) #789.6

print(LR_test_rmse) #717.2


#-------------------------------------------------------#

#              (2) decision tree              #


library(rpart)

DT_model = rpart(cnt ~ ., data = train_data, method = "anova")

DT_model


#predicting for train and test data

predictions_DT_train= predict(DT_model,train_data[,-25])

predictions_DT_test= predict(DT_model,test_data[,-25])


# MAPE calculation
```

```
DT_train_mape = MAPE(train_data[,25],predictions_DT_train)

DT_test_mape = MAPE(test_data[,25],predictions_DT_test)


# Rsquare calculation

DT_train_r2= rsquare(train_data[,25],predictions_DT_train)

DT_test_r2 = rsquare(test_data[,25],predictions_DT_test)


# RMSE calculation

DT_train_rmse = RMSE(train_data[,25],predictions_DT_train)

DT_test_rmse = RMSE(test_data[,25],predictions_DT_test)


print(DT_train_mape) #0.522

print(DT_test_mape) #0.243

print(DT_train_r2) #0.811

print(DT_test_r2) #0.798

print(DT_train_rmse) #833.848

print(DT_test_rmse) #885.59


#-------------------------------------------------------#

#              (3) Random Forest             #


#building random forest model
```

```
RF_model = randomForest(cnt~.,data = train_data,n.trees = 600)
print(RF_model)

#lets predict for both train and test data
predictions_RF_train = predict(RF_model,train_data[-25])
predictions_RF_test = predict(RF_model,test_data[-25])

#MAPE calculation
RF_train_mape = MAPE(predictions_RF_train,train_data[,25])
RF_test_mape = MAPE(predictions_RF_test,test_data[,25])

#Rsquare calculation
RF_train_r2 = rsquare(predictions_RF_train,train_data[,25])
RF_test_r2 = rsquare(predictions_RF_test,test_data[,25])

#RMSE calculation
RF_train_rmse = RMSE(train_data[,25],predictions_RF_train)
RF_test_rmse = RMSE(test_data[,25],predictions_RF_test)

print(RF_train_mape) #0.07
print(RF_test_mape) #0.12
print(RF_train_r2) #0.965
```

```
print(RF_test_r2) #0.910

print(RF_train_rmse) #371.18

print(RF_test_rmse) #593.84


#----------------------------------------------------#

#                model selection                 #


Model_name = c('Linear regression',

           'Decision tree',

           'Random forest')


MAPE_train = c(LR_train_mape,DT_train_mape,

           RF_train_mape)


MAPE_test = c(LR_test_mape,DT_test_mape,

          RF_test_mape)


Rsquare_train = c(LR_train_r2,DT_train_r2,

             RF_train_r2)


Rsquare_test = c(LR_test_r2,DT_test_r2,

             RF_test_r2)
```

```
RMSE_train =  c(LR_train_rmse,DT_train_rmse,
            RF_train_rmse)

RMSE_test = c(LR_test_rmse,DT_test_rmse,
           RF_test_rmse)

FINAL_RESULTS = data.frame(Model_name,MAPE_train,MAPE_test,Rsquare_train,Rsquare_test,
                    RMSE_train,RMSE_test)

print(FINAL_RESULTS)
```

```
#Index  Model_name        MAPE_train  MAPE_test   Rsquare_train  Rsquare_test  RMSE_train  RMSE_test
#1     Linear regression   0.15497164  0.1829289   0.8311816    0.8671739    789.6785   717.2833
#2     Decision tree       0.52210598  0.2438791   0.8119266    0.7986807    833.4855   885.5906
#3     Random forest       0.07256787  0.1224177   0.9652738    0.9103488    371.1827   593.8403

# Based on the above inferences,we came to know that Random forest performs very well in our dataset
#so we are finalising that model.
```

Python Code:

```
#import working libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
get_ipython().magic('matplotlib inline')
import statsmodels.api as sm

from scipy.stats import chi2_contingency
from statsmodels.formula.api import ols
from sklearn.linear_model import LinearRegression
from sklearn.cross_validation import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn import metrics
```

#set working directory

```
os.chdir(r"C:\Users\himanshu gupta\Desktop\edwisor\project\2")
```

#load data

```
bike_rental_data= pd.read_csv("day.csv")
```

 # ### data exploration

```
#dimension of data
bike_rental_data.shape

#checking first 5 rows
bike_rental_data.head(5)

#checking data type of all variables
bike_rental_data.dtypes
```

```
# checking summary of the dataset

bike_rental_data.describe()


#converting some useful variables into categorical variabels

categorical_var= ['season','yr','mnth','holiday','weekday','workingday','weathersit']

for a in categorical_var:

   bike_rental_data[a]=bike_rental_data[a].astype("category")


#checking datatypes again

bike_rental_data.dtypes



# we will not use instant,dateday, casual and registered variable because they are not caryying useful information.


# ### data preprocessing


# ##### target variable distribution


fig,(ax1,ax2) = plt.subplots(ncols=2)

fig.set_size_inches(10,6)
```

```python
sn.distplot(bike_rental_data["cnt"],ax=ax1)
stats.probplot(bike_rental_data["cnt"], dist='norm', fit=True, plot=ax2)

# we can cleary see that cnt is very close to normal distribution.

# ##### missing value analysis

bike_rental_data.isnull().sum()

# there are no missing values.

# ##### outliner analysis

# from above boxplot following things are clear:

fig, axes = plt.subplots(nrows=2,ncols=2)
fig.set_size_inches(14,14)
sn.boxplot(data=bike_rental_data,y="cnt",orient='v',ax=axes[0][0])
sn.boxplot(data=bike_rental_data,y="cnt",x="season",orient='v',ax=axes[0][1])
sn.boxplot(data=bike_rental_data,y="cnt",x="weekday",orient="v",ax=axes[1][0])
sn.boxplot(data=bike_rental_data,y="cnt",x="workingday",orient="v",ax=axes[1][1])
```

```
axes[0][0].set(ylabel='cnt',title = "Boxplot of cnt")

axes[0][1].set(xlabel="season",ylabel="cnt",title="Boxplot for cnt vs season")

axes[1][0].set(xlabel="weekday", ylabel="cnt",title="Boxplot for cnt vs weekday")

axes[1][1].set(xlabel="workingday",ylabel="cnt",title="Boxplot for cnt vs workingday")


# (1)there are no outliers in count.

# (2)demands for bike is very low in spring season.


# from above boxplot following things are clear:

# (1)there are no outliers in count.

# (2)demands for bike is very low in spring season.


fig, axes = plt.subplots(nrows=2,ncols=2)

fig.set_size_inches(14,14)

sn.boxplot(data=bike_rental_data,y="cnt",x="yr",orient='v',ax=axes[0][0])

sn.boxplot(data=bike_rental_data,y="cnt",x="mnth",orient='v',ax=axes[0][1])

sn.boxplot(data=bike_rental_data,y="cnt",x="holiday",orient='v',ax=axes[1][0])

sn.boxplot(data=bike_rental_data,y="cnt",x="weathersit",orient='v',ax=axes[1][1])

axes[0][0].set(xlabel="yr",ylabel="cnt",title="Boxplot for cnt vs yr")

axes[0][1].set(xlabel="mnth",ylabel="cnt",title="Boxplot for cnt vs mnth")

axes[1][0].set(xlabel="holiday",ylabel="cnt",title="Boxplot for cnt vs holiday")
```

```
axes[1][1].set(xlabel="weathersit",ylabel="cnt",title="Boxplot for cnt vs weathersit")
```

```
# from above boxplot following things are clear:

# (1) demands for bike is high in year 2011.
# (2) demands for bike is gardually incaresing from january to september and then started to decreasing.
# (3) demands for bike is high when there is clear weather.
```

```
fig, axes = plt.subplots(nrows=2,ncols=2)
fig.set_size_inches(14,14)
sn.boxplot(data=bike_rental_data,y="temp",orient='v',ax=axes[0][0])
sn.boxplot(data=bike_rental_data,y="atemp",orient='v',ax=axes[0][1])
sn.boxplot(data=bike_rental_data,y="hum",orient='v',ax=axes[1][0])
sn.boxplot(data=bike_rental_data,y="windspeed",orient='v',ax=axes[1][1])
axes[0][0].set(ylabel="temp",title="Boxplot for temp")
axes[0][1].set(ylabel="atemp",title="Boxplot for atemp")
axes[1][0].set(ylabel="hum",title="Boxplot for hum")
axes[1][1].set(ylabel="windspeed",title="Boxplot for windspeed")
```

```
# From the above boxplot we can cleary see that there are:

# (1) outliers in windspeed.

# (2) inliers in humidity.


#removal of outliers and inliers

cnames=["hum","windspeed"]

for i in cnames:

    print(i)

    q75, q25 = np.percentile(bike_rental_data.loc[:,i], [75 ,25])

    iqr = q75 - q25

    min = q25 - (iqr*1.5)

    max = q75 + (iqr*1.5)

    print(min)

    print(max)

    bike_rental_data = bike_rental_data.drop(bike_rental_data[bike_rental_data.loc[:,i] < min].index)

    bike_rental_data = bike_rental_data.drop(bike_rental_data[bike_rental_data.loc[:,i] > max].index)

    min = bike_rental_data.loc[bike_rental_data[i] < min,i]

    max = bike_rental_data.loc[bike_rental_data[i] > max,i]
```

subsituted inliers with minimum values and outliers with maximum values.

#checking humidity and windspeed after removal of inliers and outliers

fig.set_size_inches(14,14)

sn.boxplot(data=bike_rental_data,y="hum").set_title("Boxplot of humidity")

fig.set_size_inches(14,14)

sn.boxplot(data=bike_rental_data,y="windspeed").set_title("Boxplot of windspeed")

 # ##### feature selection

##Correlation analysis

#numeric variables

cnames=["temp","atemp","hum","windspeed","cnt"]

#Correlation plot

bike_rental_corr = bike_rental_data.loc[:,cnames]

#Set the width and hieght of the plot

f, ax = plt.subplots(figsize=(9, 7))

```
#Generate correlation matrix

corr = bike_rental_corr.corr()


#Plot using seaborn library

sn.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), 
cmap=sn.diverging_palette(220, 10, as_cmap=True),

            square=True, ax=ax)



# from the above plot,we came to know that both temp and atemp variables are carrying almost same information

# hence there is no need to continue with both variables.so we need to drop any one of the variables

# here we are dropping atemp variable.


#Anova test for categorical variables(target variable is numeric)

#Save categorical variables

cat_names = ["season", "yr", "mnth", "holiday", "weekday", "workingday", "weathersit"]


for i in cat_names:

    results = ols('cnt' + '~' + i, data = bike_rental_data).fit()
```

```
aov_table = sm.stats.anova_lm(results, typ = 2)
print(aov_table)
```

```
# based on the anova result, we are going to drop three variables holiday,weekday,workingday
# because these variables have the p-value > 0.05

# Removing the variables which have p-value > 0.05 and are correlated variable or does not contain useful information and store into a new dataset
df = bike_rental_data.drop(['atemp', 'holiday','weekday','workingday','instant','dteday','casual','registered'], axis=1)
bike_rental_data=df.copy()

#now check dimension of data
bike_rental_data.shape
bike_rental_data
```

 # ##### feature scaling

```
col=["temp","hum","windspeed","cnt"]
```

```
for i in col:
    print(i)
    sn.distplot(bike_rental_data[i],bins='auto',color='black')
    plt.title("distribution plot for "+i)
    plt.ylabel("density")
    plt.show()
```

based on distribution plot we can clearly see that all the numeric variables are normalized.

 # ##### bivariate analysis

Bivariate analysis of cnt and continous variables

```
fig,(ax1,ax2,ax3) = plt.subplots(ncols=3)
fig.set_size_inches(12,8)

sn.regplot(x="temp",y="cnt",data=bike_rental_data,ax=ax1)
sn.regplot(x="hum",y="cnt",data=bike_rental_data,ax=ax2)
```

```
sn.regplot(x="windspeed",y="cnt",data=bike_rental_data,ax=ax3)
```

from above boxplot it is clear that bike count has:

(1) positive linear relationship with temperature.

(2) slightly negative linear relationship with humidity.

(3) negative linear reltionship with windspeed.

 # # model development

#In Regression problems, we can't directly pass categorical variables.so we need to convert all categorical variables

#into dummy variables.

```
ccol=['season','yr','mnth','weathersit']
```

Converting categorical variables to dummy variables

```
df = pd.get_dummies(bike_rental_data,columns=ccol)
bike_rental_data=df
```

#Divide the data into train and test set

```
x= bike_rental_data.drop(['cnt'],axis=1)
y= bike_rental_data['cnt']
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=.25)

# Function for Error metrics to calculate the performance of model
def MAPE(y_true,y_prediction):
    mape= np.mean(np.abs(y_true-y_prediction)/y_true)*100
    return mape
```

 # ### linear regression model

```
LinearRegression_model= sm.OLS(y_train,x_train).fit()
print(LinearRegression_model.summary())

# Model prediction on  train data
LinearRegression_train= LinearRegression_model.predict(x_train)

# Model prediction on test data
LinearRegression_test= LinearRegression_model.predict(x_test)
```

```
# Model performance on train data
MAPE_train= MAPE(y_train,LinearRegression_train)

# Model performance on test data
MAPE_test= MAPE(y_test,LinearRegression_test)

# r2 value for train data
r2_train= r2_score(y_train,LinearRegression_train)

# r2 value for test data-
r2_test=r2_score(y_test,LinearRegression_test)

# RMSE value for train data
RMSE_train = np.sqrt(metrics.mean_squared_error(y_train,LinearRegression_train))

# RMSE value for test data
RMSE_test = np.sqrt(metrics.mean_squared_error(y_test,LinearRegression_test))

print("Mean Absolute % Error for train data="+str(MAPE_train))
print("Mean Absolute % error for test data="+str(MAPE_test))
print("R^2_score for train data="+str(r2_train))
```

```
print("R^2_score for test data="+str(r2_test))
print("RMSE for train data="+str (RMSE_train))
print("RMSE for test data="+str(RMSE_test))

Error_MetricsLT = {'Model Name': ['Linear Regression'],
          'MAPE_Train':[MAPE_train],
          'MAPE_Test':[MAPE_test],
          'R-squared_Train':[r2_train],
          'R-squared_Test':[r2_test],
          'RMSE_train':[RMSE_train],
          'RMSE_test':[RMSE_test]}

LinearRegression_Results = pd.DataFrame(Error_MetricsLT)
LinearRegression_Results
```

 # ### random forest

```
# Random Forest for regression
RF_model= RandomForestRegressor(n_estimators=80).fit(x_train,y_train)
```

```
# Prediction on train data
RF_train= RF_model.predict(x_train)

# Prediction on test data
RF_test= RF_model.predict(x_test)

# MAPE For train data
MAPE_train= MAPE(y_train,RF_train)

# MAPE For test data
MAPE_test= MAPE(y_test,RF_test)

# Rsquare  For train data
r2_train= r2_score(y_train,RF_train)

# Rsquare  For test data
r2_test=r2_score(y_test,RF_test)

# RMSE value for train data
RMSE_train = np.sqrt(metrics.mean_squared_error(y_train,RF_train))
```

```
# RMSE value for test data
RMSE_test = np.sqrt(metrics.mean_squared_error(y_test,RF_test))

print("Mean Absolute % Error for train data="+str(MAPE_train))
print("Mean Absolute % Error for test data="+str(MAPE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
print("RMSE for train data="+str (RMSE_train))
print("RMSE for test data="+str(RMSE_test))

Error_MetricsRF = {'Model Name': ['Random Forest'],
             'MAPE_Train':[MAPE_train],
             'MAPE_Test':[MAPE_test],
             'R-squared_Train':[r2_train],
             'R-squared_Test':[r2_test],
             'RMSE_train':[RMSE_train],
             'RMSE_test':[RMSE_test]}

RandomForest_Results = pd.DataFrame(Error_MetricsRF)
RandomForest_Results
```

```
# ### decision tree

# Decision tree for regression
DecisionTree_model= DecisionTreeRegressor(max_depth=3).fit(x_train,y_train)

# Model prediction on train data
DecisionTree_train= DecisionTree_model.predict(x_train)

# Model prediction on test data
DecisionTree_test= DecisionTree_model.predict(x_test)

# Model performance on train data
MAPE_train= MAPE(y_train,DecisionTree_train)

# Model performance on test data
MAPE_test= MAPE(y_test,DecisionTree_test)

# r2 value for train data
r2_train= r2_score(y_train,DecisionTree_train)

# r2 value for test data
```

```
r2_test=r2_score(y_test,DecisionTree_test)

# RMSE value for train data
RMSE_train = np.sqrt(metrics.mean_squared_error(y_train,DecisionTree_train))

# RMSE value for test data
RMSE_test = np.sqrt(metrics.mean_squared_error(y_test,DecisionTree_test))

print("Mean Absolute Precentage Error for train data="+str(MAPE_train))
print("Mean Absolute Precentage Error for test data="+str(MAPE_test))
print("R^2_score for train data="+str(r2_train))
print("R^2_score for test data="+str(r2_test))
print("RMSE for train data="+str(RMSE_train))
print("RMSE for test data="+str(RMSE_test))

Error_MetricsDT = {'Model Name': ['Decision Tree'],
        'MAPE_Train':[MAPE_train],
        'MAPE_Test':[MAPE_test],
        'R-squared_Train':[r2_train],
        'R-squared_Test':[r2_test],
        'RMSE_train':[RMSE_train],
```

```
            'RMSE_test':[RMSE_test]}

DecisionTree_Results = pd.DataFrame(Error_MetricsDT)
DecisionTree_Results
```

From above results Random Forest & linear regression both model have optimum values and this algorithms are good for our data

#saving the out put of finalized model (random forest)

```
input = y_test.reset_index()
predicted = pd.DataFrame(RF_test,columns = ['predicted'])
Final_result = predicted.join(input)
Final_result

Final_result.to_csv("Final_results_python.csv",index=False)
```

References:

1) Edwisor Learning
2) www.geeksforgeeks.org
3) towardsdatascience.com
4) rbloggers.com
5) Kaggle.com