Bike Rental

Author-

Pooja Yadwani

Contents

Chapter 1 3

Introduction 3

1.1 Problem Statement 3

1.2 Data 3

Chapter 2 5

Methodology 5

2.1 Exploratory Data Analysis (EDA) 5

2.1.1 Target Variable - cnt 6

2.1.2 Uniqueness in Variable 6

2.1.3 Missing Value Analysis 7

2.1.5 Outlier Analysis 9

2.2 Feature Selection 11

2.3 Dummy Variables 12

2.4 Feature Scaling 13

Chapter 3 14

Modeling 14

3.1 Model Selection 14

3.3 Model Building 14

3.3.1 Decision Tree 14

3.3.2 Random Forest 15

3.3.3 Linear Regression 15

3.3.4 Gradient boosting 15

Chapter 4 16

Conclusion 16

4.1 Model Evaluation 16

4.2 Model Selection 17

Appendix A: Extra Figures 18

Appendix B: R Code 21

References 36

Chapter 1

Introduction

Bike sharing is a brilliant idea which provides people with another short-range transportation option that allows them to travel without worrying about being stuck in traffic and maybe enjoy city view or even workout at the same.

In this project, we will be investigating into the bike share rental data from "Capital Bikeshare" servicing Washington D.C. and surrounding areas beginning 2010. Capital Bikeshare was the largest bike sharing service in the United States when they started, until Citi Bike for New York City started operations in 2013. Capital Bikeshare started from 10 stations and 120 bicycles in Washington D.C. and expanding into a bike share system that owns more than 429 stations and 2500 bicycles and also services Arlington County, Fairfax County, City of Alexandria in Virginia and Montgomery County in Maryland.

Objective of the analysis is to find out the determining factor that drives the demand on bike share rentals, construct statistical models and then try to make prediction on rentals based on the information and models we have. Exploration and the analysis of the data will be performed in R and Python.

1.1 Problem Statement

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

1.2 Data

Bike_Rental dataset was provided for analysis. Data contains 15 predictor variables and 1 target variable.

Variables	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)
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	(extracted from Freemeteo)
Temp	Normalized temperature in Celsius
Atemp	Normalized feeling temperature in Celsius.
Hum	Normalized humidity
Windspeed	Normalized wind speed
Casual	count of casual users
Registered	count of registered users
Cnt	count of total rental bikes including both casual and registered

Size of Dataset Provided: - 731 rows, 16 Columns

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp
1	40544	1	0	1	0	6	0	2	0.344167	0.363625
2	40545	1	0	1	0	0	0	2	0.363478	0.353739
3	40546	1	0	1	0	1	1	1	0.196364	0.189405
4	40547	1	0	1	0	2	1	1	0.2	0.212122
5	40548	1	0	1	0	3	1	1	0.226957	0.22927

hum	windspeed	casual	registered	cnt
0.805833	0.160446	331	654	985
0.696087	0.248539	131	670	801
0.437273	0.248309	120	1229	1349
0.590435	0.160296	108	1454	1562
0.436957	0.1869	82	1518	1600

Notes:

- '**cnt'** is our target variable
- '**cnt'** is sum of two variables – '**registered'** and '**casual'**. So, we have excluded the **both (casual** and **registered)** columns.
- **'instant'** variable is of no use and can be dropped
- **'dteday'** variable is a date column which is not significant in our analysis and can be excluded

Chapter 2

Methodology

Bike Rental is a business scenario in which a company is trying to analyze reasons for employees talking frequent day off from work. For reducing absenteeism rate, we need to identify the issues which employees face or reasons they present to the firm while talking day off. Also, we have some data to train our model which makes our problem as Supervised Classification problem.

- **Exploratory Data Analysis (EDA)-** It includes following steps
  - Looking into the data and analyzing all variables
  - Visualization
  - Missing Value Analysis
  - Outlier Analysis
  - Correlation analysis
  - Feature Scaling
  - Dummy data creation
  - Feature Sampling.

- **Basic Modeling-** Trying different models over preprocessed data
  - Decision Tree
  - Random forest
  - Linear regression
  - Gradient Boosting

- **Model Evaluation & Optimization-** Evaluating model performances and then selecting the best model fit for our data, optimizing hyper parameters tuning and cost effectiveness of model. This step is optional. We may or may not involve it. It is basically done to avoid a scenario where the selected approach works very well with training data but fails to support out test data in similar way.

- **Implementation model on Final test data and saving the results**

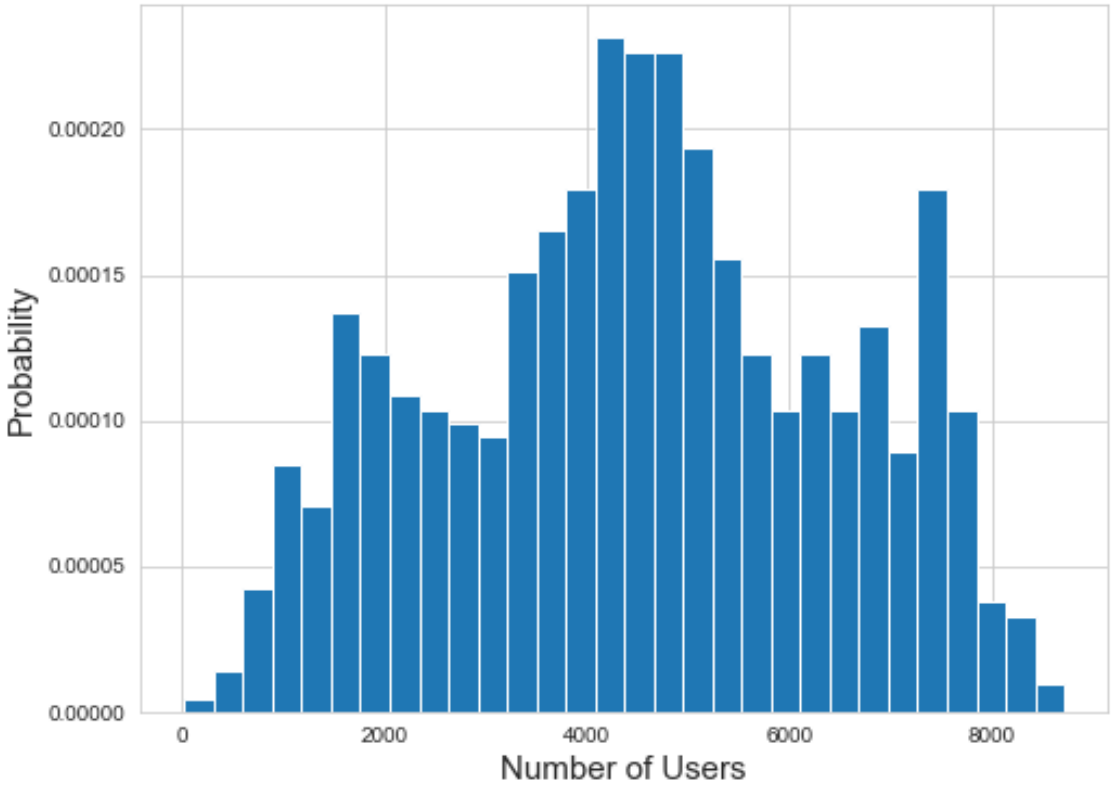
2.1 Exploratory Data Analysis (EDA)

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before getting them dirty with it.

To start this process, we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions of the variable.

2.1.1 Target Variable - cnt

Our target variable is continuous which includes 696 unique values. This variable shows the number of users who have rented a bike on same day.



2.1.2 Uniqueness in Variable

In the given data set there are 16 variables and data types of all variables are either float64 or int64 and one is object type i.e Date column. There are 731 observations and 16 columns in our data set. Missing value is not present in our data. Below is the count of Unique values for each column.

Variable	Unique Counts
Instant	731
Dteday	731
Season	4
yr	2
mnth	12
holiday	2
weekday	7
workingday	2
weathersit	3
temp	499
atemp	690

hum	595
windspeed	650
casual	606
registered	679
cnt	696

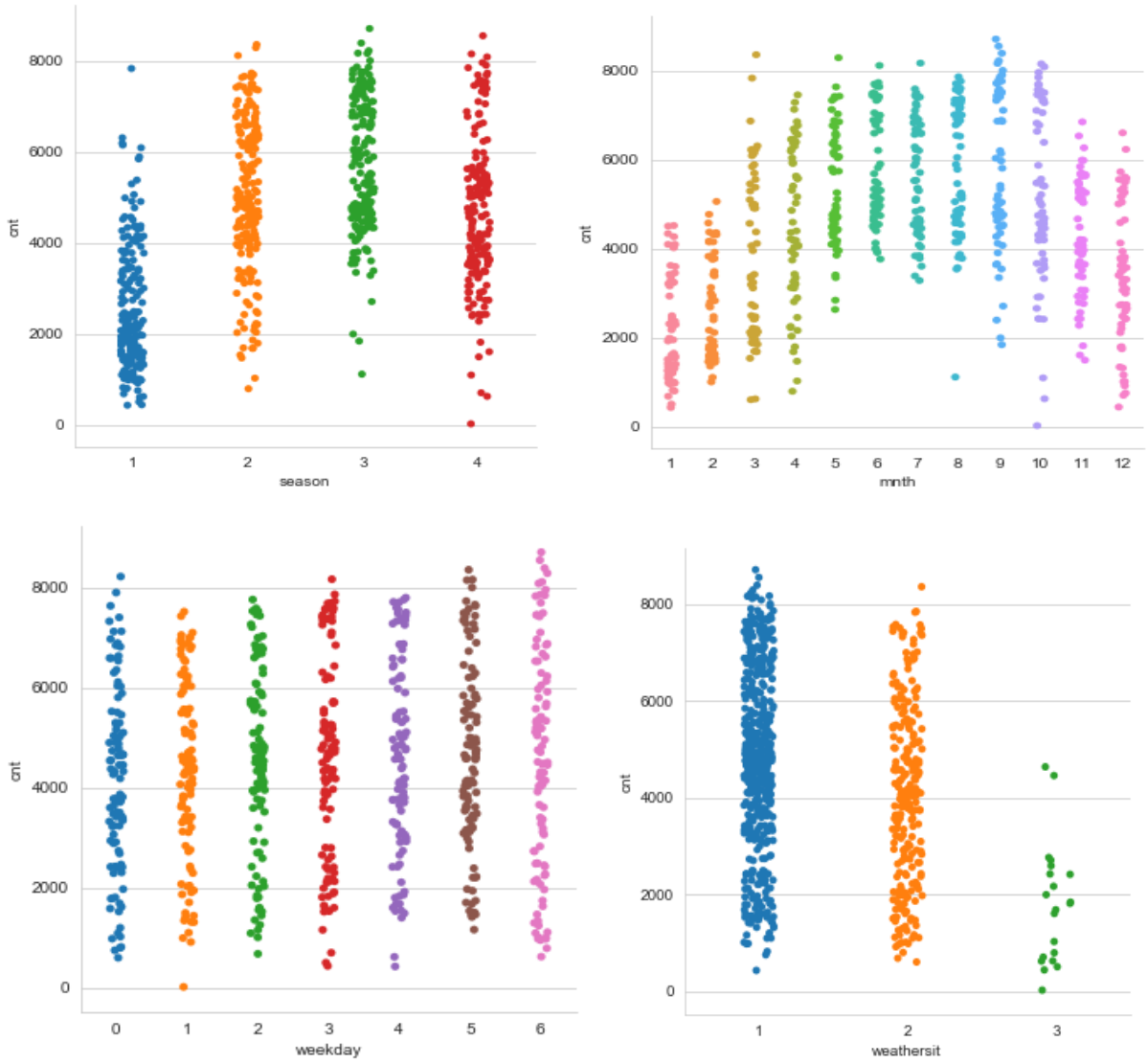
2.1.3 Missing Value Analysis

In statistics, *missing data*, or *missing values*, occur when no *data value* is stored for the variable in an observation. *Missing data* are a common occurrence and can have a significant effect on the conclusions that can be drawn from the *data*. If a column has more than 30% of data as missing value either we ignore the entire column, or we ignore those observations. In the given data we have no missing values for any variable.

Variables	# of Null Values
Instant	0
Dteday	0
Season	0
Yr	0
Mnth	0
Holiday	0
Weekday	0
Workingday	0
Weathersit	0
Temp	0
Atemp	0
Hum	0
Windspeed	0
Casual	0
Registered	0
Cnt	0

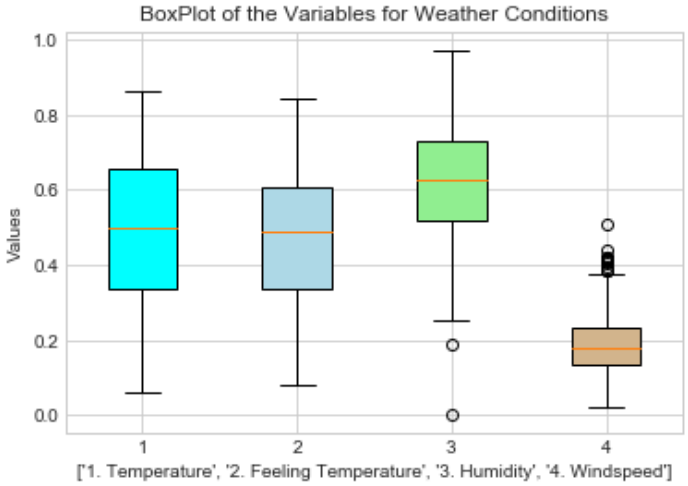
2.1.4 Data Understanding

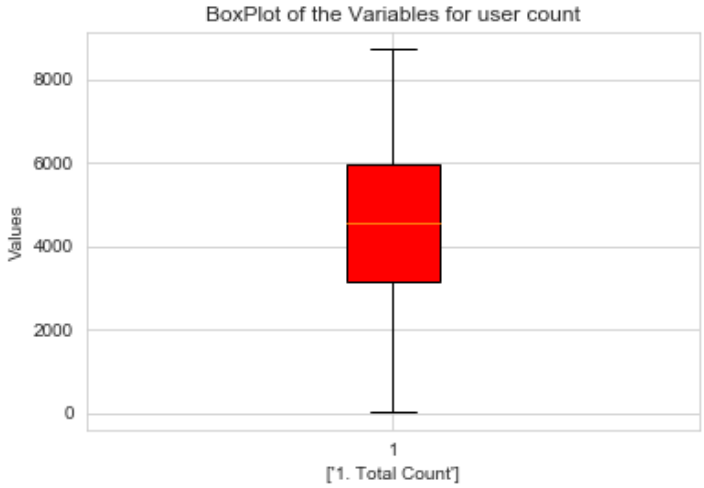
In order to get further insight and understand the data set and to see how different features interact with each other and the target. First the amount of bike rental counts for each day of the week is analyzed.



2.1.5 Outlier Analysis

We can clearly observe from these probability distributions that most of the variables are skewed. The skew in these distributions can be most likely explained by the presence of outliers and extreme values in the data. One of the other steps of pre-processing apart from checking for normality is the presence of outliers. In this case we use a classic approach of removing outliers.
We visualize the outliers using boxplots.





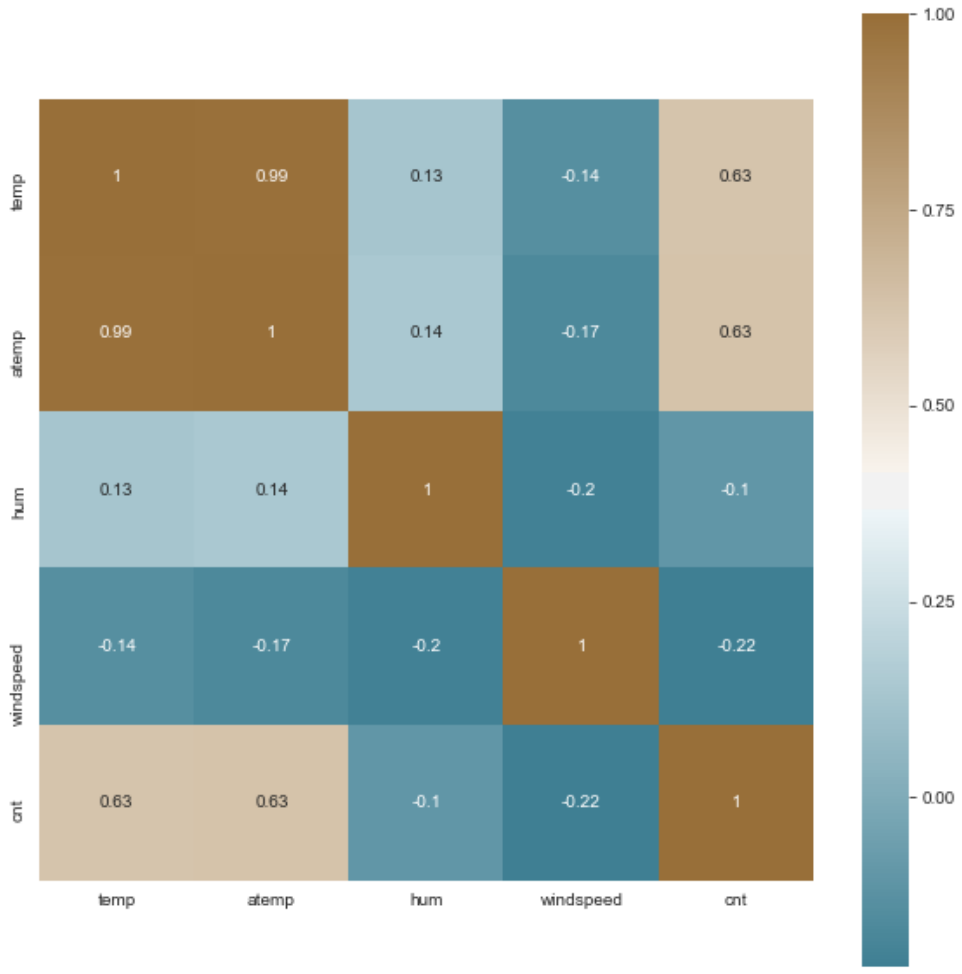
From the boxplot only **"casual" and "windspeed"** variables consist of outliers.

To deal with outliers:

- **Windspeed** - We have converted the outliers (data beyond minimum and maximum values) as NA i.e. missing values and calculated % for null values. It was 1.778% which is very less and so we can delete the rows with nulls (previously outliers) for the column 'windspeed' or we can impute. Here we have imputed with mean.
- **Casual** – We have not done any imputation for this variable as it was to be removed in next step as part of Feature Selection since it is highly correlated with other variable.

2.2 Feature Selection

Before performing any type of modeling, we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. Selecting subset of relevant columns for the model construction is known as **Feature Selection**. We cannot use all the features because some features may be carrying the same information or irrelevant information which can increase overhead. To reduce overhead, we adopt feature selection technique to extract meaningful features out of data. This in turn helps us to avoid the problem of multi collinearity. In this project we have selected **Correlation Analysis** for numerical variable and **ANOVA** (Analysis of variance) for categorical variable.



From correlation analysis and ANOVA test we have found that

- **'temp'** and '**atemp'** have high correlation (>0.7), so we have excluded the **atemp** column.
- '**holiday**', '**weekday**' and '**workingday**' have p>0.05 and hence were excluded.

2.3 Dummy Variables

A Dummy variable or Indicator Variable is an artificial variable created to represent an attribute with two or more distinct categories/levels. Dummies are any variables that are either one or zero for each observation. pd.get_dummies when applied to a column of categories where we have **one** category per observation will produce a new column (variable) for each unique categorical value. It will place a one in the column corresponding to the categorical value present for that observation.
This is equivalent to one hot encoding.

One-hot encoding is characterized by having only one per set of categorical values per observation.

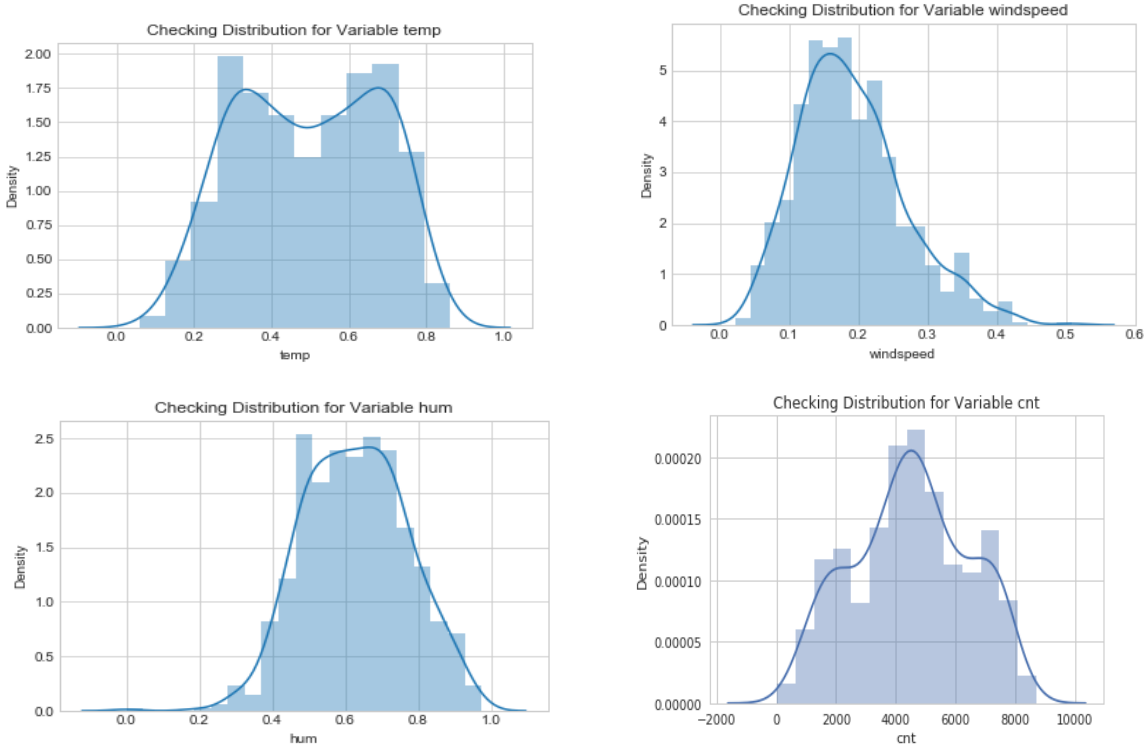
Viewing data after adding dummy variables:

	temp	hum	windspeed	cnt	season_1	season_2	season_3	season_4	yr_0	yr_1	...	mnth_6	mnth_7	mnth_8	mnth_9	mnth_10	mnth_11	mnth_12	weathersit_1	weathersit_2	weathersit_3
0	-0.826097	1.249316	-0.364668	985	1	0	0	0	1	0	...	0	0	0	0	0	0	0	0	1	0
1	-0.720601	0.478785	0.873479	801	1	0	0	0	1	0	...	0	0	0	0	0	0	0	0	1	0
2	-1.633538	-1.338358	0.870246	1349	1	0	0	0	1	0	...	0	0	0	0	0	0	0	1	0	0
3	-1.613675	-0.263001	-0.366777	1562	1	0	0	0	1	0	...	0	0	0	0	0	0	0	1	0	0
4	-1.466410	-1.340576	0.007143	1600	1	0	0	0	1	0	...	0	0	0	0	0	0	0	1	0	0

5 rows × 25 columns

2.4 Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. Using these variables without standardization in effect gives the variable with the larger range a weight of 100 in the analysis. Transforming the data to comparable scales can prevent this problem. Typical data standardization procedures equalize the range and/or data variability. Standardized variables are the parts that must remain the same to avoid muddying the results, because if they aren't controlled, it would be less clear whether the changes to the independent variable caused the changes in the dependent variable.



Since our data is uniformly distributed we will use Standardization in this step as Feature Scaling Method.

Chapter 3

Modeling

After a thorough preprocessing we will use some regression models on our processed data to predict the target variable.

3.1 Model Selection

It has been noted in previous stages of our analysis that for different combinations of the independent variables, the count is different. The dependent variable is a continuous variable and hence the type model of model that would be developed for this problem is a regression model.

3.2 Methodology:

Model Evaluation is an integral part of the model development process as it helps us find the best model for representing our data. It also helps to evaluate as to how it would on new data. In order to develop an efficient and accurate model to predict our target variable we shall use a combination of three different methods, the three different methods that can be used are given below.

- Hold-Out Method
- Cross-Validation Technique

3.2.1 Hold-Out Method

As evaluating model performance on training data set may lead to develop an over fitted model. Due to this is required to test the model on a separate data set. Hence the original data set is split into training and testing data. The training data set is used to build a predictive model and the testing data is used to evaluate the model performance.

3.2.2 Cross Validation Technique

Cross-validation is a statistical method used to estimate the skill of machine learning models. That k-fold cross validation is a procedure used to estimate the skill of the model on new data. There are common tactics that you can use to select the value of k for your dataset.

3.3 Model Building

We Start building our model by using the following models-

3.3.1 Decision Tree

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Each branch connects nodes with “and” and multiple branches are connected by “or”. It can be used for classification and regression. It is a supervised machine learning algorithm. Accept continuous and categorical variables as independent variables. Extremely easy to understand by the business users. Split of decision tree is seen in the below tree. The MAPE, MSE value and R^2 value for our project in R and Python are –

Decision Tree	R	PYTHON
MAPE	23.37	25.71
MSE	759495.1	994781.61
R^2	0.8	0.71

3.3.2 Random Forest

Random Forest is an ensemble technique that consists of many decision trees. The idea behind Random Forest is to build n number of trees to have more accuracy in dataset. It is called random forest as we are building n no. of trees randomly. In other words, to build the decision trees it selects randomly n no of variables and n no of observations to build each decision tree. It means to build each decision tree on random forest we are not going to use the same data. The MAPE, MSE value and R^2 value for our project in R and Python are –

Random Forest	R	PYTHON
MAPE	14.17	13.33
MSE	299731.6	322023.66
R^2	0.92	0.90

3.3.3 Linear Regression

Linear Regression is one of the statistical methods of prediction. It is applicable only on continuous data. To build any model we have some assumptions to put on data and model. Here are the assumptions to the linear regression model. The MAPE, MSE value and R^2 value for our project in R and Python are –

Linear Regression	R	PYTHON
MAPE	15.09	17.21
MSE	516594.5	541997.40
R^2	0.87	0.84

3.3.4 Gradient boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The MAPE, MSE value and R^2 value for our project in R and Python are –

Gradient boosting	R	PYTHON
MAPE	14.73	13.03
MSE	370693	331509.92
R^2	0.9	0.90

Chapter 4

Conclusion

In this chapter we are going to evaluate our models, select the best model for our dataset and try to get answers of the asked questions.

4.1 Model Evaluation

In the previous chapter we have seen the **Mean Absolute Percentage Error** (MAPE), **Mean Square Error** (MSE) and **R-Squared** Value of different models.
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. In other words, it tells you how concentrated the data is around the line of best fit.
Whereas **R-squared** is a relative measure of fit which tells us the percentage of variance of our target variable explained by independent variables.
RMSE is an absolute measure of fit. 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.
The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a Loss function for regression problems in Machine Learning.

Lower values of **RMSE and MAPE** and higher value of **R-Squared Value** indicate better fit.

Model Summary:

S.No	Code	Model_name	MSE	RMSE	MAPE	R^2
1	Python	**Decision tree default**	994781.61	997.39	25.71	0.71
2		**Random Forest Default**	322023.66	567.47	13.33	0.91
3		**Linear Regression**	541997.4	736.2	17.22	0.84
4		**Gradient Boosting Default**	331509.92	575.77	13.04	0.9
5	R	**Decision tree default**	759495.1	871.49	23.37	0.8
6		**Random Forest Default**	299731.6	547.48	14.17	0.92
7		**Linear Regression**	516594.5	718.75	15.09	0.87
8		**Gradient Boosting Default**	370693	608.85	14.73	0.9

4.2 Model Selection

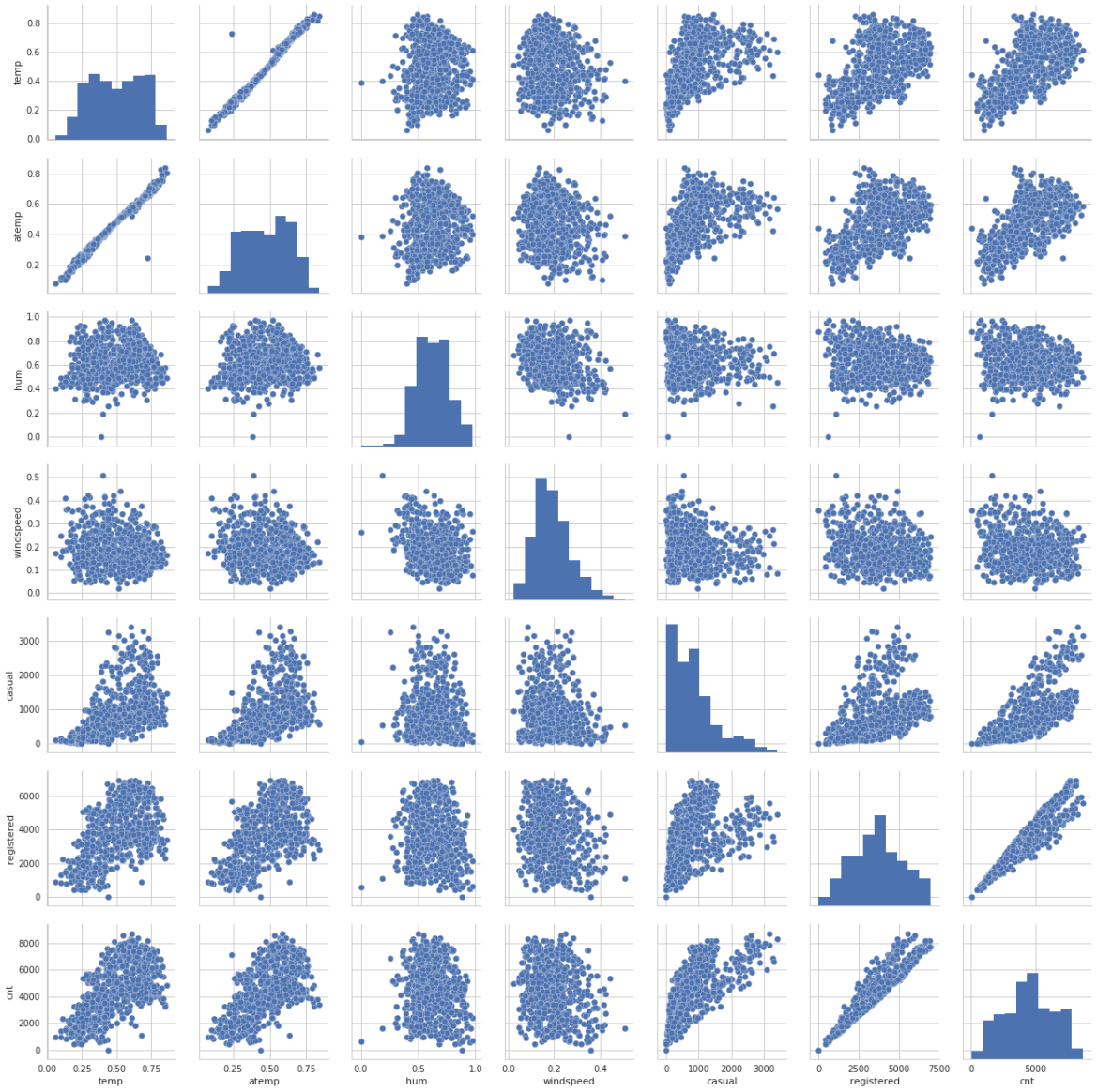
From the observation of all **MAPE, MSE Value** and **R-Squared** Value we have concluded that, Both the models- **Gradient Boosting Default** and **Random Forest** perform comparatively well while comparing their MSE, R-Squared value and MAPE.
After this, I chose **Random Forest** as a method to apply cross validation technique and see changes brought about by that.

Random Search CV Random Forest Regressor Model Performance:
Best Parameters = {'n_estimators': 15, 'max_depth': 23}
R-squared = 0.89.
MSE = 381271
MAPE = 14.11%

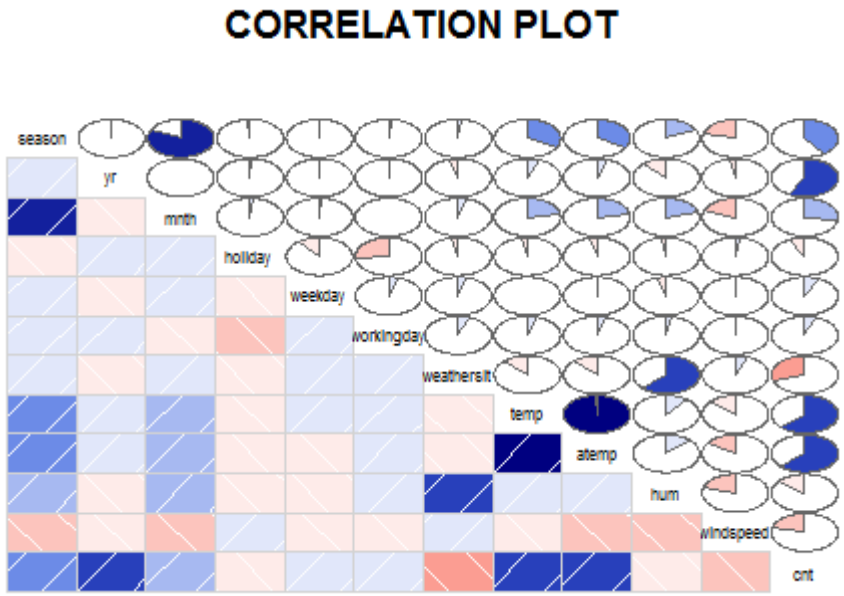
Conclusion:
We can conclude that there is no significant change in our result after applying CV technique for optimization. This shows that our model was generating best results.

Appendix A: Extra Figures

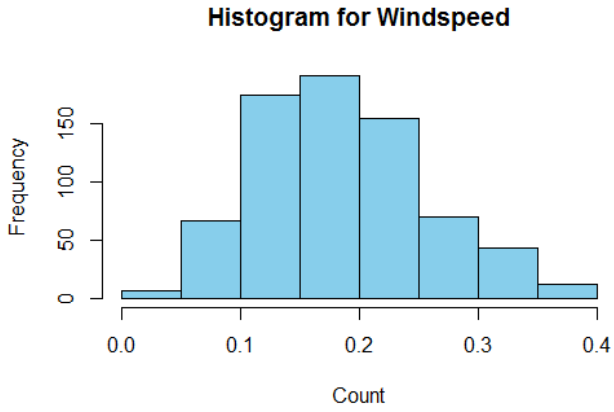
Pairplot

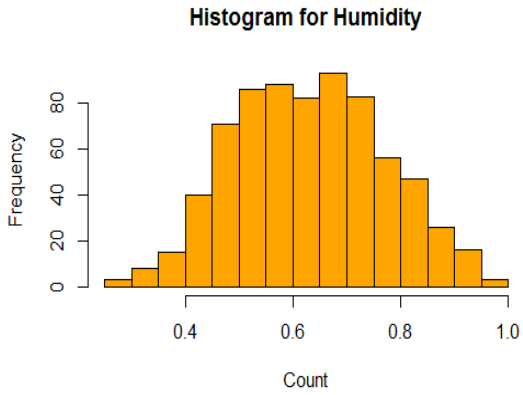


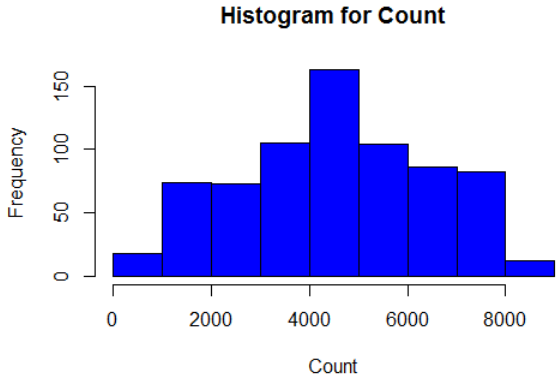
Corrgram Plot in R

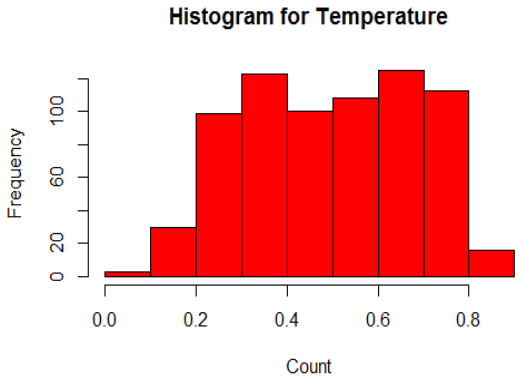


Histogram

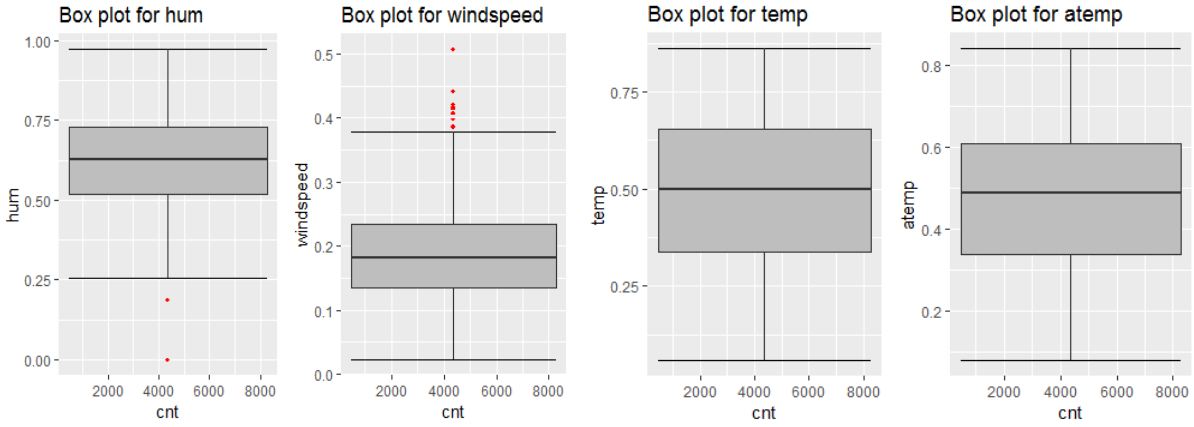




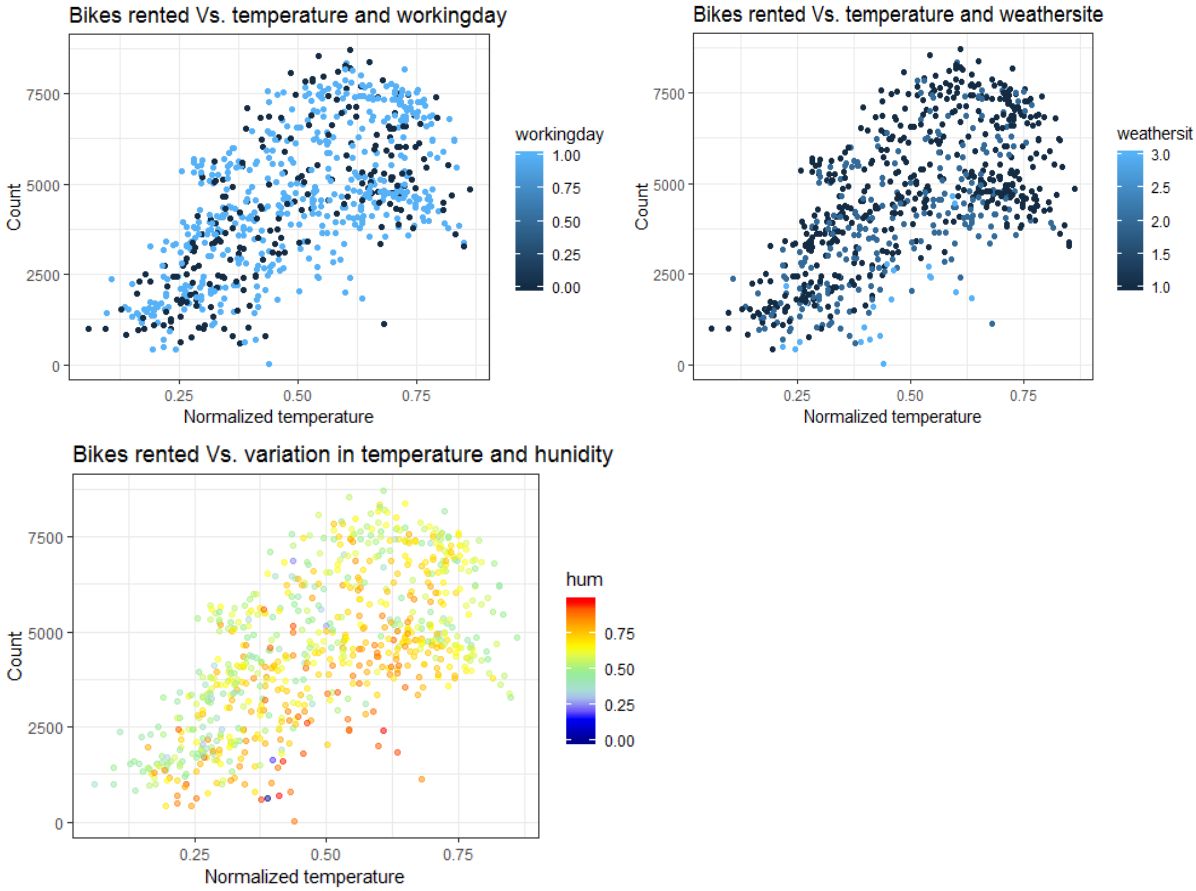




Boxplot



Scatter Plots in R



Appendix B: R Code

```
rm(list = ls())

setwd("C:/Users/Click/Desktop/Bike rental")

getwd()

# #loading Libraries

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "e1071",

      "DataCombine", "pROC", "doSNOW", "class", "readxl","ROSE","dplyr", "plyr",
"reshape","xlsx","pbapply", "unbalanced", "dummies", "MASS" , "gbm" ,"Information", "rpart",
"miscTools")


# #install.packages if not

#lapply(x, install.packages)


# #load libraries

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

rm(x)


#Input Data Source

df = data.frame(read_xlsx('Bike_Rental.xlsx', sheet = 1))


#Creating backup of orginal data

data_Original  = df



#############################################################################

#               EXPLORING DATA
            #

#############################################################################


#viewing the data

head(df,5)
```

```
dim(df)

#structure of data or data types
str(df)

#Summary of data
summary(df)

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

#By looking at data we can conclude that
#variables 'instant' and 'dteday' are not significant for our analysis.
#and variables 'casual' and 'registered' sum up to target variable 'cnt'.
#Hence, we can remove these before proceeding.
df = subset(df, select = -c(casual, registered, instant, dteday))

dimnames(df)

# From the above EDA and problem statement categorising data in 2 category "continuous" and "catagorical"
cont_vars = c('temp', 'atemp', 'hum', 'windspeed', 'cnt')

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



#############################################################################
#                 Visualizing the data                      #
#############################################################################
```

```
#library(ggplot2)


#Plot Number of bikes rented Vs. the days of the week.
ggplot(data = df, aes(x = reorder(weekday,-cnt), y = cnt))+
  geom_bar(stat = "identity")+
  labs(title = "Number of bikes rented Vs. days", x = "Days of the week", y = "Count")+
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))



#Plot Bikes rented Vs. variation in temperature and hunidity
#It can be observed that people rent bikes mostly when temperature in between 0.5 and0.75 normalized temperature
#and between normalized humidity 0.50 and 0.75
ggplot(df,aes(temp,cnt)) +
  geom_point(aes(color=hum),alpha=0.5)+
  labs(title = "Bikes rented Vs. variation in temperature and hunidity", x = "Normalized temperature", y = "Count")+
  scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red')) +
  theme_bw()

#Plot Bikes rented Vs. temperature and weathersite
#Most bikes are rented duing weather site forcast 1
ggplot(data = df, aes(x = temp, y = cnt))+
  geom_point(aes(color=weathersit))+
  labs(title = "Bikes rented Vs. temperature and weathersite", x = "Normalized temperature", y = "Count")+
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
  theme_bw()

#PLot Bikes rented Vs. temperature and workingday
#People rent bikes mostly on working weekdays
```

```
ggplot(data = df, aes(x = temp, y = cnt))+
  geom_point(aes(color=workingday))+
  labs(title = "Bikes rented Vs. temperature and workingday", x = "Normalized temperature",y = "Count")+
  #  theme(panel.background = element_rect("white"))+
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
  theme_bw()



#####################################################################
#                                Missing data                                                  #
#####################################################################

missing_val = sum(is.na(df))
print(missing_val)



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

## BoxPlots - Distribution and Outlier Check
# Boxplot for continuous variables
for (i in 1:length(cont_vars))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cont_vars[i]), x = "cnt"), data = subset(df))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
```

```
        labs(y=cont_vars[i],x="cnt")+
        ggtitle(paste("Box plot for",cont_vars[i])))
}

# ## Plotting plots together
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,ncol=2)
#gridExtra::grid.arrange(gn5)

# #Remove outliers using boxplot method
# #loop to remove from all variables
for(i in cont_vars)
{
  print(i)
  val = df[,i][df[,i] %in% boxplot.stats(df[,i])$out]
  #print(length(val))
  df = df[which(!df[,i] %in% val),]
}

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

# Imputing missing values
df = knnImputation(df,k=3)
```

```
#####################################################################
#               Feature Selection                                   #
#####################################################################

#Here we will use corrgram to find corelation

##Correlation plot
#library('corrgram')

corrgram(df,
         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 that the highly corr related vars in plot are marked in dark blue.
#Dark blue color means highly positive correlation

##------------------ANOVA testset--------------------------##

## ANOVA testset for Categprical variable
summary(aov(formula = cnt~season,data = df))
summary(aov(formula = cnt~yr,data = df))
summary(aov(formula = cnt~mnth,data = df))
summary(aov(formula = cnt~holiday,data = df))
summary(aov(formula = cnt~weekday,data = df))
summary(aov(formula = cnt~workingday,data = df))
```

```
summary(aov(formula = cnt~weathersit,data = df))



#####################################################################
#             Feature Selection                                     #
#####################################################################

## Dimension Reduction
#temp and atemp have high correlation (>0.7), so we have excluded the atemp column.
#holiday, weekday, workingday have p>0.05

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

#####################################################################
#             Feature Scaling                                       #
#####################################################################

# Updating the continuous and catagorical variables

cont_vars = c('temp', 'hum', 'windspeed', 'cnt')

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



#Normality check
#Checking Data for Continuous Variables

################  Histogram  ##################
hist(df$cnt, col="blue", xlab="Count", main="Histogram for Count")
hist(df$hum, col="orange", xlab="Count", main="Histogram for Humidity")
```

```
hist(df$windspeed, col="sky blue", xlab="Count", main="Histogram for Windspeed")
hist(df$temp, col="red", xlab="Count", main="Histogram for Temperature")

#We have seen that our data is mostly normally distributed. Hence, we will go for Standardization.

#Viewing data before Standardization.
head(df)
cont_vars

#Standardization
for(i in cont_vars)
{
  if(i!= "cnt"){
    print(i)
    df[,i] = (df[,i] - mean(df[,i]))/(sd(df[,i]))
  }
}

#Viewing data after Standardization.
head(df)

#Creating dummy variables for categorical variables
library(mlr)
df1 = dummy.data.frame(df, cata_vars)

#Viewing data after adding dummies
head(df1)

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

```
#####################################################################

# #Divide data into trainset and testset using stratified sampling method

#install.packages('caret')
#library(caret)
set.seed(101)
split_index = createDataPartition(df1$cnt, p = 0.8, list = FALSE)
trainset = df1[split_index,]
testset  = df1[-split_index,]

#Checking df Set Target Class
table(trainset$cnt)



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



#############################################################################################
#########################################################################
##                                  Basic approach for ML - Models

##
##            We will first get a basic idea of how different models perform on our preprocesed data and
then select the best model and make it       ##
##                                   more efficient for our Dataset
                                                                                    ##
#############################################################################################
#########################################################################
```

```
#----------------------------------------Decision tree----------------------------------------#
#Develop Model on training data
fit_DT = rpart(cnt ~., data = trainset, method = "anova")

#Variable importance
fit_DT$variable.importance

#      temp        yr     season       mnth        hum  windspeed weathersit
# 889796479  637857947  548442480  487529776  195102361  135504939   32013972

#Lets predict for test data
pred_DT_test = predict(fit_DT, testset)

# For test data
print(postResample(pred = pred_DT_test, obs = testset$cnt))

#Compute R^2
dt_r2 = rSquared(testset$cnt, testset$cnt - pred_DT_test)
print(dt_r2)

#Compute MSE
dt_mse = mean((testset$cnt - pred_DT_test)^2)
print(dt_mse)

#Compute MAPE
dt_mape = MAPE(testset$cnt, pred_DT_test)
print(dt_mape)

#RMSE  Rsquared       MAE
```

```
#871.4901885   0.8050212 672.9650742

#R2
#0.8044511

#MSE
#759495.1

#MAPE
#23.36856



#----------Linear Regression----------#

#Develop Model on training data
fit_LR = lm(cnt ~ ., data = trainset)

#Lets predict for test data
pred_LR_test = predict(fit_LR, testset)

# For test data
print(postResample(pred = pred_LR_test, obs = testset$cnt))

#Compute R^2
lr_r2 = rSquared(testset$cnt, testset$cnt - pred_LR_test)
print(lr_r2)

#Compute MSE
lr_mse = mean((testset$cnt - pred_LR_test)^2)
print(lr_mse)
```

```
#Compute MAPE
lr_mape = MAPE(testset$cnt, pred_LR_test)
print(lr_mape)

#RMSE   Rsquared        MAE
#718.7451054   0.8690861 532.9476425

#R2
#0.8669913

#MSE
#516594.5

#MAPE
#15.08846

#----------------------------------------Random Forest----------------------------------------------#

#Develop Model on training data
fit_RF = randomForest(cnt~., data = trainset)

#Lets predict for test data
pred_RF_test = predict(fit_RF, testset)

# For test data
print(postResample(pred = pred_RF_test, obs = testset$cnt))

#Compute R^2
rf_r2 = rSquared(testset$cnt, testset$cnt - pred_RF_test)
```

```
print(rf_r2)

#Compute MSE
rf_mse = mean((testset$cnt - pred_RF_test)^2)
print(rf_mse)

#Compute MAPE
rf_mape = MAPE(testset$cnt, pred_RF_test)
print(rf_mape)

#     RMSE   Rsquared        MAE
# 547.4775243   0.9258038 422.5625912

#R2
#0.9228274

#MSE
#299731.6

#MAPE
#14.17132

#----------------------------------------XGBoost----------------------------------------#

#Develop Model on training data
fit_XGB = gbm(cnt~., data = trainset, n.trees = 500, interaction.depth = 2)

#Lets predict for test data
pred_XGB_test = predict(fit_XGB, testset, n.trees = 500)
```

```
# For test data
print(postResample(pred = pred_XGB_test, obs = testset$cnt))

#Compute R^2
xgb_r2 = rSquared(testset$cnt, testset$cnt - pred_XGB_test)
print(xgb_r2)

#Compute MSE
xgb_mse = mean((testset$cnt - pred_XGB_test)^2)
print(xgb_mse)

#Compute MAPE
xgb_mape = MAPE(testset$cnt, pred_XGB_test)
print(xgb_mape)

#      RMSE   Rsquared        MAE
#608.8456581   0.9045705 468.6529898

#R2
#0.9045568

#MSE
#370693

#MAPE
#14.73772

##################----------Viewing summary of all models----------###############
# Create variables
```

```
Model_name <- c("Decision Tree","Linear Regression", "Random Forest", "XGBoost" )
MSE <- c(dt_mse, lr_mse, rf_mse, xgb_mse)
r2 <- c(dt_r2, lr_r2, rf_r2, xgb_r2)
MAPE <- c(dt_mape, lr_mape, rf_mape, xgb_mape)

# Join the variables to create a data frame
results <- data.frame(Model_name,MSE,r2,MAPE)
results

#     Model_name      MSE        r2     MAPE
#1     Decision Tree 759495.1 0.8044511 23.36856
#2 Linear Regression 516594.5 0.8669913 15.08846
#3     Random Forest 299731.6 0.9228274 14.17132
#4           XGBoost 370693.0 0.9045568 14.73772

##############################################################################################
###
#                                        Saving output to file
                                  #
##############################################################################################
###

#swrite.csv(submit,file = 'C:/Users/Click/Desktop/Bike rental/Finalcount_R.csv',row.names = F)
#rm(list = ls())
```

References

- ✓ https://seaborn.pydata.org/generated/seaborn.pairplot.html
- ✓ https://www.programcreek.com/python/example/96210/seaborn.pairplot
- ✓ https://stats.stackexchange.com/questions/321085/f-value-of-zero-in-anova
- ✓ https://stackoverflow.com/questions/32244753/how-to-save-a-seaborn-plot-into-a-file
- ✓ https://medium.com/@eijaz/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f
- ✓ https://en.wikipedia.org/wiki/Hyperparameter_optimization
- ✓ https://www.guru99.com/r-decision-trees.html
- ✓ https://stackoverflow.com/questions/10438752/adding-x-and-y-axis-labels-in-ggplot2
- ✓ https://docs.python.org/3.4/library/statistics.html
- ✓ https://seaborn.pydata.org/generated/seaborn.heatmap.html
- ✓ https://www.researchgate.net/post/what_are_the_best_methods_for_filling_in_missing_values
- ✓ https://scikit-learn.org/0.15/auto_examples/imputation.html
- ✓ https://scikit-learn.org/0.15/auto_examples/imputation.html
- ✓ https://datascience.stackexchange.com/questions/24989/imputing-for-multiple-missing-variables-using-sklearn
- ✓ https://datascience.stackexchange.com/questions/24989/imputing-for-multiple-missing-variables-using-sklearn