# Predicting Rental Bikes

Lakshmi Amrutha Sai

January 2020

# Contents

**1 Introduction**
1.1 Problem Statement
1.2 Data

**2 Methodology**
2.1 Pre Processing
2.1.1 Exploratory Data Analysis
2.1.2 Missing Value Analysis
2.1.3 Outlier Analysis
2.1.4 Feature Selection
2.1.4 Feature Scaling
2.2 Modeling
2.2.1 Model Selection
2.2.2 Multiple Linear Regression
2.2.3 Regression Trees
2.2.4 Classification

**3 Conclusion**
3.1 Model Evaluation
3.1.1 Mean Absolute Error (MAPE)
3.2 Model Selection

**Appendix A - R Code**

**Appendix B - Python Code**

# 1. INTRODUCTION

## 1.1 Problem statement

A bike rental is a bicycle business that rents bikes for short periods of time. Most rentals are provided by bike shops as a sideline to their main businesses of sales and service, but some shops specialize in rentals. Bike rental shops rent by the day or week as well as by the hour, and these provide an excellent opportunity for people who don't have access to a vehicle, typically travelers and particularly tourists. Specialized bike rental shops thus typically operate at beaches, parks, or other locations that tourists frequent. In this case, the fees are set to encourage renting the bikes for a few hours at a time, rarely more than a day. The objective of this Case is to predict the bike rental count based on the environmental and seasonal settings, So that required bikes would be arranged and managed by the shops according to environmental and seasonal conditions.

## 1.2 Data

In this project, our task is to build regression models which will used to predict the bike rental count on daily basis. Given below is a sample of the bike sharing dataset:

Table 1.1: Bike sharing Dataset (Columns: 1-8)

| instant | dteday | Season | Yr | mnth | holiday | weekday |
|---|---|---|---|---|---|---|
| 1 | 1/1/2011 | 1 | 0 | 1 | 0 | 6 |
| 2 | 1/2/2011 | 1 | 0 | 1 | 0 | 0 |
| 3 | 1/3/2011 | 1 | 0 | 1 | 0 | 1 |
| 4 | 1/4/2011 | 1 | 0 | 1 | 0 | 2 |
| 5 | 1/5/2011 | 1 | 0 | 1 | 0 | 3 |

Table 1.2: Bike Rental Sample Data (Columns: 9-14)

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

From the table below we have the following 16 variables, using which we have to predict the bike rental count:

Table 1.3: Predictor Variables

| SL. No. | Predictor |
|---|---|
| **1** | record id |
| 2 | datetime |
| **3** | season |
| 4 | year |
| **5** | month |
| 6 | holiday |
| **7** | weekday |
| 8 | workingday |
| **9** | weather_condition |
| 10 | temp |
| **11** | atemp |
| 12 | humidity |
| **13** | windspeed |
| 14 | casual |
| **15** | registered |
| 16 | total_count |

# 2. Methodology

## 2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

### 2.1.1 Exploratory Data Analysis

In exploring the data we have

- Converted season, mnth, workingday, weathersit into categorical variables
- Feature Engineering :Changed deday variables's date value to day of date and converted to categorical variable having 31 levels as a month has 31 days.
- Deleted instant variable as it is nothing but an index.
- Omitted registered and casual variable as sum of registered and casual is the total count that is what we have to predict.

### 2.1.2 Missing Value Analysis

Missing value analysis is done to check is there any missing value present in given dataset. Missing values can be easily treated using various methods like mean, median method, knn method to impute missing value.

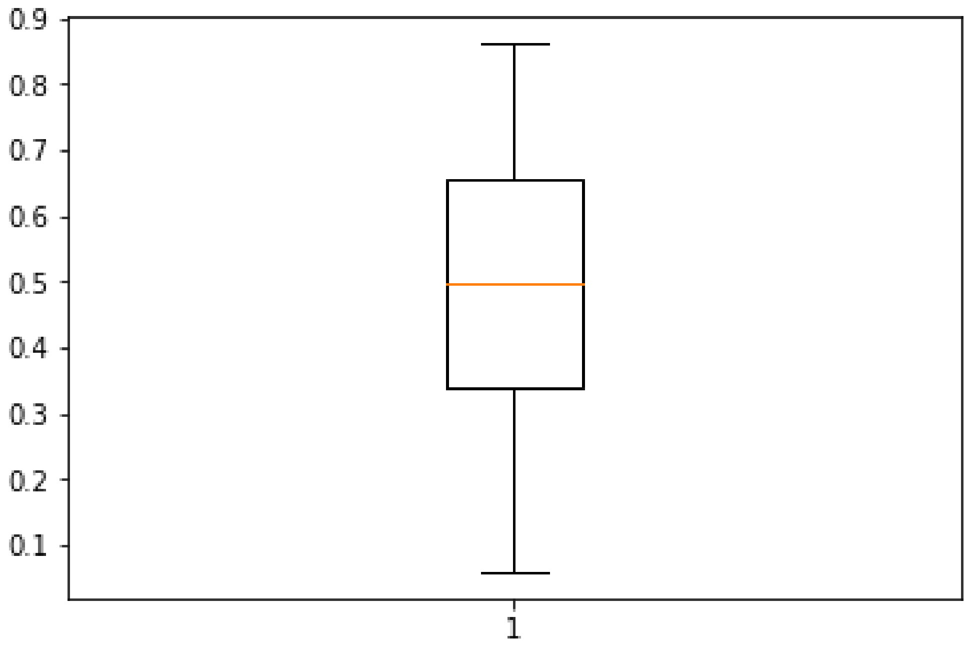
```
#Missing values in dataset R code
missing_val = data.frame(apply(bike_df,2,function(x){sum(is.na(x))}))
names(missing_val)[1]='missing_val'

# Python code
df_day.isnull().sum()
```

| s. no | Variables | missing values |
|---|---|---|
| 1 | dteday | 0 |
| 2 | season | 0 |
| 3 | yr | 0 |
| 4 | mnth | 0 |
| 5 | holiday | 0 |
| 6 | weekday | 0 |
| 7 | workingday | 0 |
| 8 | weathersit | 0 |
| 9 | temp | 0 |
| 10 | atemp | 0 |
| 11 | hum | 0 |
| 12 | windspeed | 0 |
| 13 | casual | 0 |
| 14 | registered | 0 |

There are no missing value found in given dataset.

### 2.1.3 Outlier Analysis

Outlier analysis is done to handle all inconsistent observations present in given dataset. As outlier analysis can only be done on continuous variable.
Figure 2.1 and 2.2 are visualization of numeric variable present in our dataset to detect outliers using boxplot.



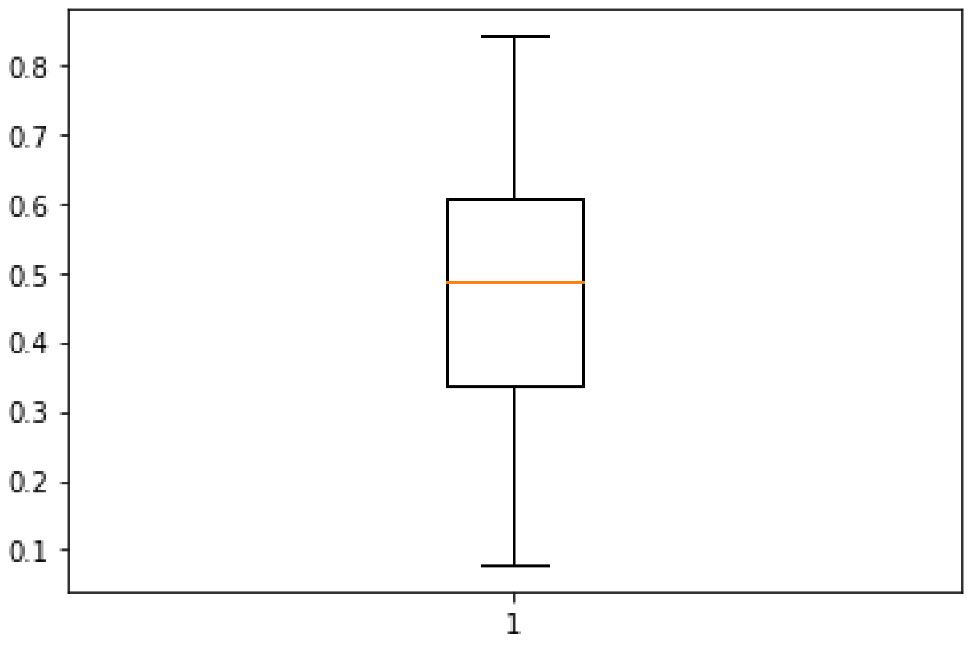


Figure 2.1 Boxplot graph of temp and atemp variables

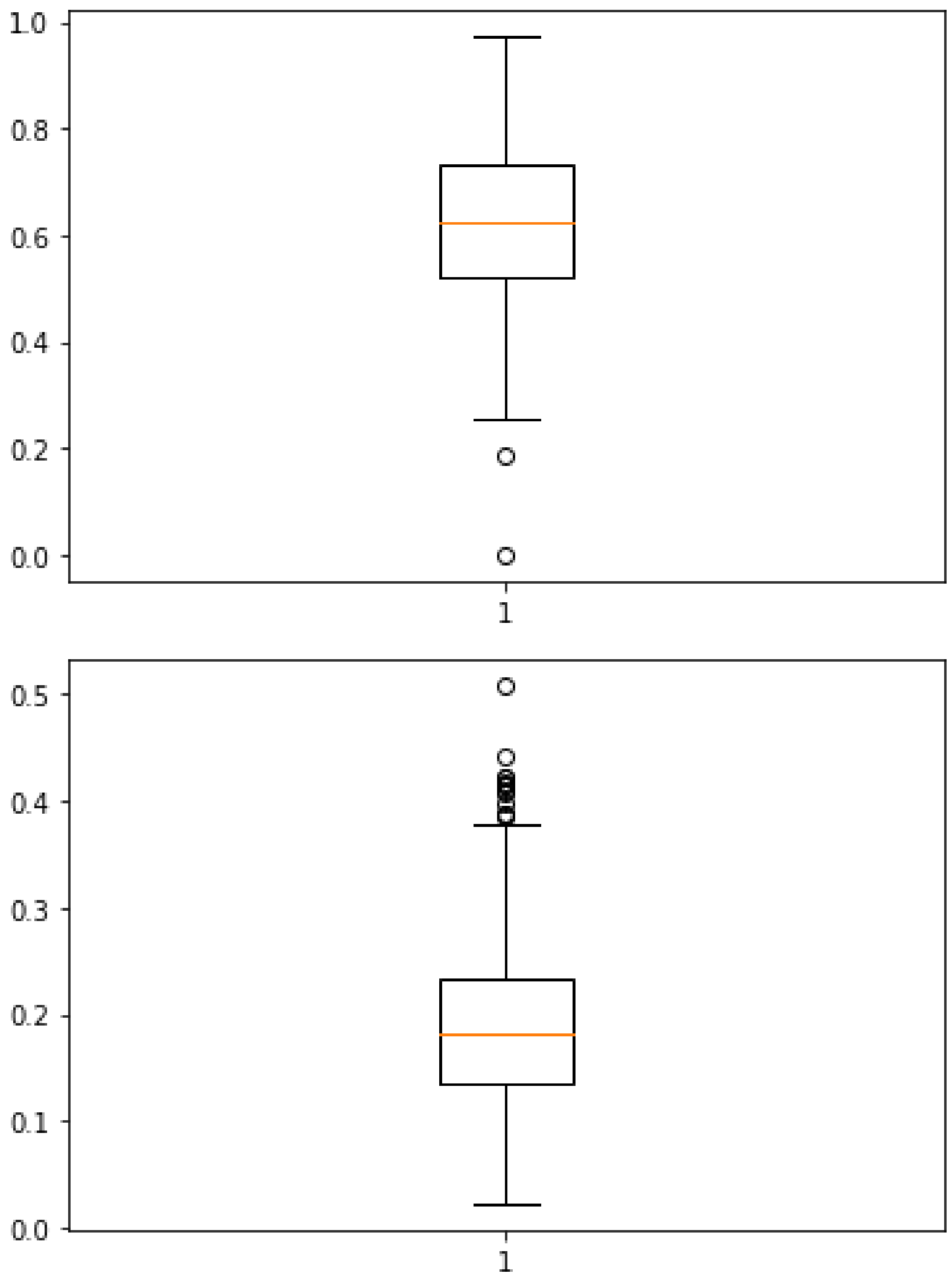


Figure 2.2 Boxplot graph of hum and windspeed variables

According to above visualizations there is no outlier found in temp and atemp variable but there are few outliers found in windspeed and hum variable.
As windspeed variable defines the windspeed on a particular day and hum defines the humidity of that day so we can neglect these outliers because both these variable define environmental condition. Due to drastic change in weather like strome, heavy rain condition.

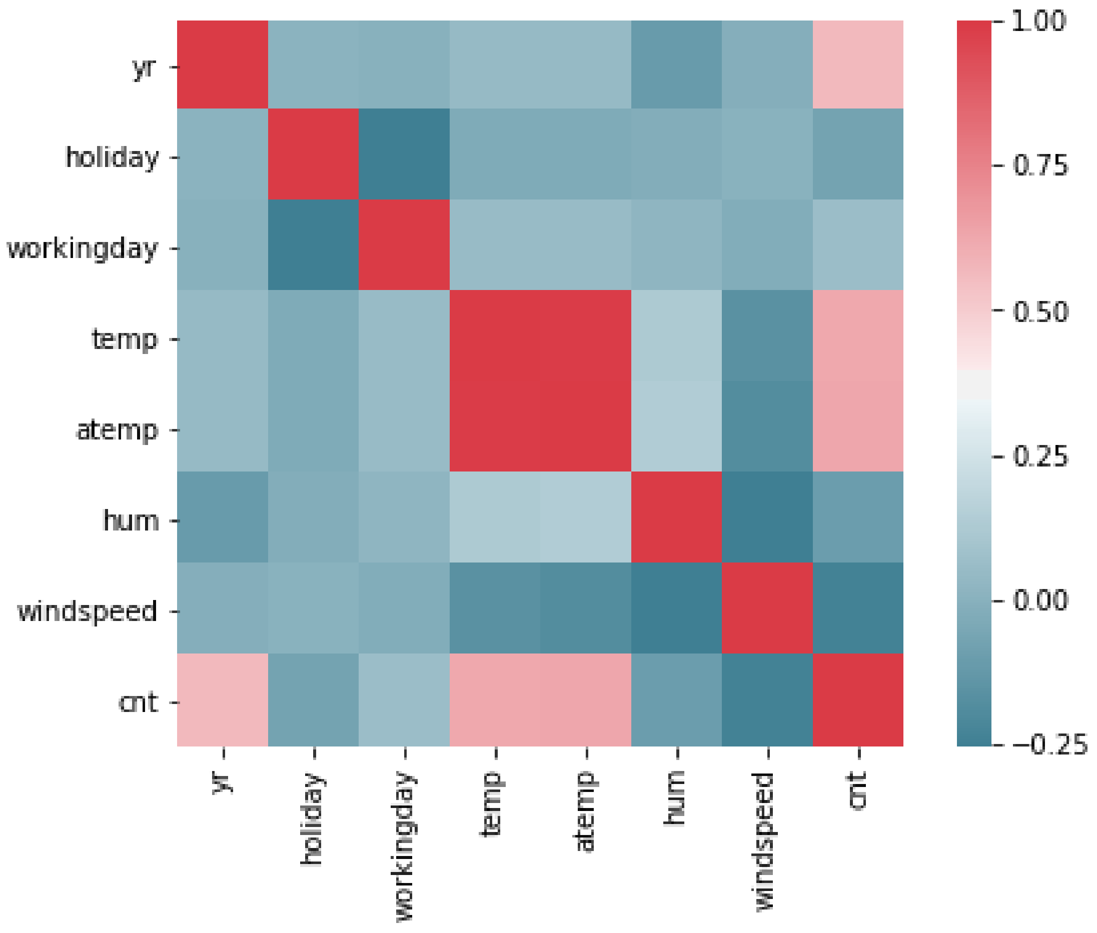
## 2.1.4 Feature Selection

Feature selection is very important for modelling the dataset. The every dataset have good and unwanted features. The unwanted features will effect on performance of model, so we have to delete those features. We have to select best features by using ANOVA, Chi-Square test and correlation matrix statistical techniques and so on. In this, we are selecting best features by using Correlation matrix.

**Correlation matrix**

Correlation matrix is tells about linear relationship between attributes and help us to build better models.

From the correlation plot, we can observed that some features are positively correlated and some are. negatively correlated to each other. The temp and atemp are highly positively correlated to each other, it means that both are carrying same information. So, we are going to ignore atemp variable for further analysis.



### 2.1.5Features Scaling

The word “normalization” is used informally in statistics, and so the term normalized data can have multiple meanings. In most cases, when you normalize data you eliminate the units of measurement for data, enabling you to more easily compare data from different places. Some of the more common ways to normalize data include:
Transforming data using a z-score or t-score. This is usually called standardization. In the vast majority of cases, if a statistics textbook is talking about normalizing data, then this is the definition of “normalization” they are probably using.
Rescaling data to have values between 0 and 1. This is usually called feature scaling. One possible formula to achieve this is.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

In rental dataset numeric variables like 'temp' , 'atem' ,'hum' and ' windspeed' are in normalization form so , we have to Normalize two variables 'casual' and 'registered' .
After normalize 'casual' and 'registered' variables look like in table below where all values between 0 and 1

Table Normalization of 'casual' and 'registered

| casual | registered |
|---|---|
| 0.037852113 | 0.09384926 |
| 0.034624413 | 0.17455963 |
| 0.025234742 | 0.21628646 |
| 0.042840376 | 0.21628646 |
| 0.019366197 | 0.12575801 |

## 2.2 Modeling

### 2.2.1 Model Selection

In this case we have to predict the count of bike renting according to environmental and seasonal condition. So the target variable here is a continuous variable. For Continuous we can use various Regression models. Model having less error rate and more accuracy will be our final model.
Models built are
1. c50 (Decision tree for regression target variable)
2. Random Forest (with 200 trees)
3. Linear regression

#### 2.2.2 C50

This model is also known a Decision tree for regression target variable.
For this model we have divided the dataset into train and test part using random sampling. Where train contains 80% data of data set and test contains 20% data and contains 12 variable where 12th variable is the target variable.
Creating Model
In R
```
fit = rpart(cnt ~ ., data = train, method = "anova")
predictions_DT = predict(fit, test[,-12])
```

In python
```
fit_DT = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:11],
train.iloc[:,11])
predictions_DT = fit_DT.predict(test.iloc[:,0:11])
```

The R-squared or coefficient of determination is 0.74 on average for 3-fold cross validation, it means that predictor is only able to predict 74% of the variance in the target variable which is contributed by all the independent variables.

### 2.2.3 Random Forest

In Random forest we have divided the dataset into train and test part using random sampling. For this model we have divided the dataset into train and test part using random sampling Where train contains 80% data of data set and test contains 20% data and contains 12 variable where 12th variable is the target variable.

In R

```
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)
predictions_RF = predict(RF_model, test[,-12])
plot(RF_model)
```

In Python

```
#random forest
RFmodel = RandomForestRegressor(n_estimators = 200).fit(train.iloc[:,0:11],
train.iloc[:,11])
RF_Predictions = RFmodel.predict(test.iloc[:,0:11])
```

### 2.2.4 Linear Regression

For linear regression model we have divided the categorical containing more than 2 classes into dummy variable. So that all categorical variable should be in binary classes form. On creating dummy variable there are 64 variable in both R and Python. Where 64th is the target variable.

Further the data is again divided into train and test with 80 % train data and 20 % test data using random sampling.

In R

```
#Linear regression model making
lm_model = lm(cnt ~., data = train_lr)
predictions_LR = predict(lm_model,test_lr[,-64])
```

In Python

```
trainlr, testlr = train_test_split(data_lr, test_size=0.2)
model = sm.OLS(trainlr.iloc[:,63], trainlr.iloc[:,0:63]).fit()
predictions_LR = model.predict(testlr.iloc[:,0:63])
```

## 3. Conclusion

### 3.1 Model Evaluation

Now, we have a three models for predicting the target variable, but we need to decide which model better for this project. There are many metrics used for model evaluation.

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

In this project, we are using three metrics for model evaluation as follows,

**1. R-squared or Coefficient of Determination ($R2$) :-** It explains the how much variance of dependent variable which is contributed by all the independent variables.

**2. Root Mean Square Error (RMSE) :-** It is the square root of the mean of the square of difference between the true and predicted dependent variable values.

**3. Mean Absolute Error (MEA) :-** It is the mean of the absolute difference between the true and predicted dependent variable values.

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

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of Bike Renting, the latter two, *Interpretability* and *Computation Efficiency*, do not hold much significance. Therefore we will use *Predictive performance* as the criteria to compare and evaluate models.

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

#### 3.1.1 Mean Absolute Percentage Error (MAPE)

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

MAPE values is R are as follow

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

```
#MAPE for decision tree regression
MAPE(test[,12], predictions_DT)

#MAPE for random forest regression
MAPE(test[,12], predictions_RF)

#MAPE for linear regression
MAPE(test_lr[,64],  predictions_LR)
```

MAPE value in Python are as follow

```
#defining MAPE function
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape
```

In above function y_true is the actual value and y_pred is the predicted value. It will provide the error percentage of model.

```
#MAPE for decision tree regression
MAPE(test.iloc[:,11], predictions_DT)
```

```
27.737837701228408
```

```
#MAPE for random forest regression
MAPE(test.iloc[:,11],RF_Predictions)
```

```
14.923072236915019
```

```
#MAPE for linear regression
MAPE(testlr.iloc[:,63], predictions_LR)
```

```
18.137949688224342
```

Where predictions_DT are predicted values from C50 model.
predictions_RF are predicted values from random forest model
predictions_LR are predicted values from linear regression model

## 3.2 Model Selection

We can see that from both R and Python Random forest model performs best out of c50 and linear regression. So random forest model is selected with 83% accuracy in R and with 86% accuracy in python.
Extracted predicted value of random forest model are saved with .csv file format.

# Appendix A - R Code

```
rm(list=ls(all=T))
setwd("C:/Users/lenovo-pc/Desktop/Data Science Projects")
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced",
"C50", "dummies", "e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine',
'inTrees','fastDummies')
lapply(x, require, character.only = TRUE)
rm(x)
bike = read.csv("day.csv", header = T, na.strings = c(" ", "", "NA"))
bike_train=bike
######Exploratory Data Analysis##################
bike_train$season=as.factor(bike_train$season)
bike_train$mnth=as.factor(bike_train$mnth)
bike_train$yr=as.factor(bike_train$yr)
bike_train$holiday=as.factor(bike_train$holiday)
bike_train$weekday=as.factor(bike_train$weekday)
bike_train$workingday=as.factor(bike_train$workingday)
bike_train$weathersit=as.factor(bike_train$weathersit)
bike_train=subset(bike_train,select = -c(instant,casual,registered))
d1=unique(bike_train$dteday)
df=data.frame(d1)
bike_train$dteday=as.Date(df$d1,format="%Y-%m-%d")
df$d1=as.Date(df$d1,format="%Y-%m-%d")
bike_train$dteday=format(as.Date(df$d1,format="%Y-%m-%d"), "%d")
bike_train$dteday=as.factor(bike_train$dteday)
str(bike_train)
###Missing Values
Analysis###############################################
# 1. checking for missing value
missing_val = data.frame(apply(bike_train,2,function(x){sum(is.na(x))}))
##############Outlier Analysis##########
# 1.BoxPlots - Distribution and Outlier Check
numeric_index = sapply(bike_train,is.numeric) #selecting only numeric
numeric_data = bike_train[,numeric_index]
cnames = colnames(numeric_data)
```

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

gridExtra::grid.arrange(gn1,gn2,ncol=3)
gridExtra::grid.arrange(gn3,gn4,ncol=2)

#####################Feature Selection################
## Correlation Plot
corrgram(bike_train[,numeric_index], order = F,
       upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
## Dimension Reduction####
bike_train = subset(bike_train,select = -c(atemp))
#######Model
Development#####################################################
#############################################
rmExcept("bike_train")
train_index = sample(1:nrow(bike_train), 0.8 * nrow(bike_train))
train = bike_train[train_index,]
test = bike_train[-train_index,]

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

set.seed(568)
#load the rpart library for decision trees
library(rpart)
#rpart.control to contro the performance of model
rpart.control<-rpart.control(minbucket = 2,cp = 0.01,maxcompete = 3,
                          maxsurrogate = 4, usesurrogate = 2, xval = 3,surrogatestyle =
0, maxdepth = 10)
```

```
#training the dtr model
dtr<-
rpart(train_encoded_attributes$total_count~.,data=train_encoded_attributes [,-
c(6)],control=rpart.control,method='anova',cp=0.01)
#load the rpart.plot for plot the learned dtr model
library(rpart.plot)
rpart.plot(dtr, box.palette="RdBu", shadow.col="gray",
nn=TRUE,roundint=FALSE)

#############Random Forest Model###########################
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)
predictions_RF = predict(RF_model, test[,-12])
plot(RF_model)
################Linear Regression#################

#converting multilevel categorical variable into binary dummy variable
cnames= c("dteday","season","mnth","weekday","weathersit")
data_lr=bike_train[,cnames]
cnt=data.frame(bike_train$cnt)
names(cnt)[1]="cnt"
data_lr <- fastDummies::dummy_cols(data_lr)
data_lr= subset(data_lr,select = -c(dteday,season,mnth,weekday,weathersit))
d3 = cbind(data_lr,bike_train)
d3= subset(d3,select = -c(dteday,season,mnth,weekday,weathersit,cnt))
data_lr=cbind(d3,cnt)
#dividind data into test and train
train_index = sample(1:nrow(data_lr), 0.8 * nrow(data_lr))
train_lr = data_lr[train_index,]
test_lr = data_lr[-train_index,]

#Linear regression model making
lm_model = lm(cnt ~., data = train_lr)
predictions_LR = predict(lm_model,test_lr[,-64])
#summary(lm_model)
#################evaluating MApe value###############
MAPE = function(y, yhat){
  mean(abs((y - yhat)/y))*100
}
MAPE(test[,12], predictions_DT)

MAPE(test[,12], predictions_RF)

MAPE(test_lr[,64],  predictions_LR)
```

```
##########extacting predicted values output from Random forest
model#######################
results <- data.frame(test, pred_cnt = predictions_RF)

write.csv(results, file = 'RF output R .csv', row.names = FALSE,
quote=FALSE)
```

# Appendix B – Python Code

```
############importing libraries#############
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform
import datetime as dt
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor
import statsmodels.api as sm
from sklearn.ensemble import RandomForestRegressor
from  matplotlib import pyplot
############working directory#################
os.chdir("C:/Users/lenovo-pc/Desktop/Data Science Projects")
###########loading dataset####################
bike_train = pd.read_csv("day.csv")
############exploratory data analysis#######################
bike_train['season']= bike_train['season'].astype('category')
bike_train['yr']=bike_train['yr'].astype('int')
bike_train['mnth']=bike_train['mnth'].astype('category')
bike_train['holiday']=bike_train['holiday'].astype('int')
bike_train['workingday']=bike_train['workingday'].astype('int')
bike_train['weekday']=bike_train['weekday'].astype('category')
bike_train['weathersit']=bike_train['weathersit'].astype('category')
d1=bike_train['dteday'].copy()
for i in range (0,d1.shape[0]):
    d1[i]=datetime.datetime.strptime(d1[i], '%Y-%m-%d').strftime('%d')
bike_train['dteday']=d1
bike_train['dteday']=bike_train['dteday'].astype('category')
bike_train = bike_train.drop(['instant','casual', 'registered'], axis=1)

bike_train.dtypes
```

```
#####################missing values################################

#total_missing_values = df_day.isnull().sum().sort_values(ascending=False)
#total_missing_value

total = bike_train.isnull().sum().sort_values(ascending=False)
percent =
(bike_train.isnull().sum()/bike_train.isnull().count()).sort_values(ascending=Fal
se)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
################Outlier Analysis###################

#saving numeric values#
cnames=["temp","atemp","hum","windspeed",]
#ploting boxplotto visualize outliers#
plt.boxplot(bike_train['temp'])
plt.boxplot(bike_train['atemp'])
plt.boxplot(bike_train['hum'])
plt.boxplot(bike_train['windspeed'])
#############Feature Selection ##################
df_corr = bike_train
#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(7, 5))
#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(7, 5))
#Generate correlation matrix
corr = df_corr.corr()
#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
#droping corelated variable
bike_train = bike_train.drop(['atemp'], axis=1)
############Modeling ################################

#dividing data into train and test
train, test = train_test_split(bike_train, test_size=0.2)
######c50#######
fit_DT = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:11],
train.iloc[:,11])
predictions_DT = fit_DT.predict(test.iloc[:,0:11])
```

```
#random forest
RFmodel = RandomFo
restRegressor(n_estimators = 200).fit(train.iloc[:,0:11], train.iloc[:,11])
RF_Predictions = RFmodel.predict(test.iloc[:,0:11])
#linear regression
#creating dummy variable
data_lr=bike_train.copy()
cat_names = ["season", "dteday", "weathersit", "mnth","weekday"]
for i in cat_names:
    temp = pd.get_dummies(data_lr[i], prefix = i)
    data_lr = data_lr.join(temp)
fields_to_drop = ['dteday', 'season', 'weathersit', 'weekday', 'mnth','cnt']
data_lr = data_lr.drop(fields_to_drop, axis=1)
data_lr=data_lr.join(bike_train['cnt'])

trainlr, testlr = train_test_split(data_lr, test_size=0.2)
model = sm.OLS(trainlr.iloc[:,63], trainlr.iloc[:,0:63]).fit()
predictions_LR = model.predict(testlr.iloc[:,0:63])
#defining MAPE function
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape
#MAPE for decision tree regression
MAPE(test.iloc[:,11], predictions_DT)
#MAPE for random forest regression
MAPE(test.iloc[:,11],RF_Predictions)
#MAPE for linear regression
MAPE(testlr.iloc[:,63], predictions_LR)

result=pd.DataFrame(test.iloc[:,0:11])
result['pred_cnt'] = (RF_Predictions)

result.to_csv("Random forest output python.csv",index=False)
```