BIKE RENTAL PREDICTION

Shridhar S

01 August 2019

INDEX

Chapter 1

Introduction

1.1 Problem Statement :

Rental market is on the rise in the modern world and Bike renting services is one of the Industries which has huge potential. At the same time good bike rental service can help people show more faith in the service. The aim of this project is to predict the Bike rental counts on a Daily basis based on the different weather conditions, seasons and other environmental conditions. This helps the company in being well prepared to offer the service depending on the factors that drive the rental counts.

1.2 Data

Our aim is to build a model which will predict the counts of rides expected in a day. We have a sample dataset given below to have an idea of how our dataset looks and what are the variables that are at out disposal in predicting the rental counts.

Table 1.1 : Bike rental sample data(columns 1-9)

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit
1	01-01-2011	1	0	1	0	6	0	2
2	02-01-2011	1	0	1	0	0	0	2
3	03-01-2011	1	0	1	0	1	1	1
4	04-01-2011	1	0	1	0	2	1	1
5	05-01-2011	1	0	1	0	3	1	1
6	06-01-2011	1	0	1	0	4	1	1
7	07-01-2011	1	0	1	0	5	1	2
8	08-01-2011	1	0	1	0	6	0	2

Table 1.2 : Bike rental sample data(columns 9-16)

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	0.089565	88	1518	1606
0.196522	0.208839	0.498696	0.168726	148	1362	1510
0.165	0.162254	0.535833	0.266804	68	891	959

As we can see from the sample data, we have a total of 16 variables in the dataset.

Table 1.3 Unique Variables

1	**instant**
2	**dteday**
3	**season**
4	**yr**
5	**mnth**
6	**holiday**
7	**weekday**
8	**workingday**
9	**weathersit**
10	**temp**
11	**atemp**
12	**hum**
13	**windspeed**
14	**casual**
15	**registered**
16	**cnt**

To give a brief introduction about the variables we can see the below description :

instant: Record index

dteday: Date

season: Season (1:sprin, 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 fromHoliday Schedule)

weekday: Day of the week

workingday: If day is neither weekend nor holiday is 1, otherwise is 0.

weathersit: (extracted fromFreemeteo)

1: Clear, Few clouds, Partly cloudy, Partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + 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_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

So in the above dataset, we can say that cnt is the dependent variable.

Chapter 2

Methodology

2.1 Pre – Processing

The data we receive would not be clean or inline with what we need for analysis. In order to make the data best for visualization, graphs and plots, we need to clean the data and make necessary changes. This is called Data Pre-Processing and it is one of the most important steps in Data Science. However, in this dataset we see that the data is mostly clean and the datapoints are may be optimized for storing purpose. Here in this project, we have replaced different features with the correct naming to be easier for analysis. For example, 1,2,3,4 is replaced by Spring, Summer, Fall and Winter. This helps us in EDA.

2.2 Missing Value Analysis

We don't see any missing values in the data. The data has all the cells filled.

2.3 Distribution of Categorical Variables

In the below graph we can see how the number of bike rentals is distributed against the categorical variables such as Season, Month, Working day, Weekday, Year, Holiday.

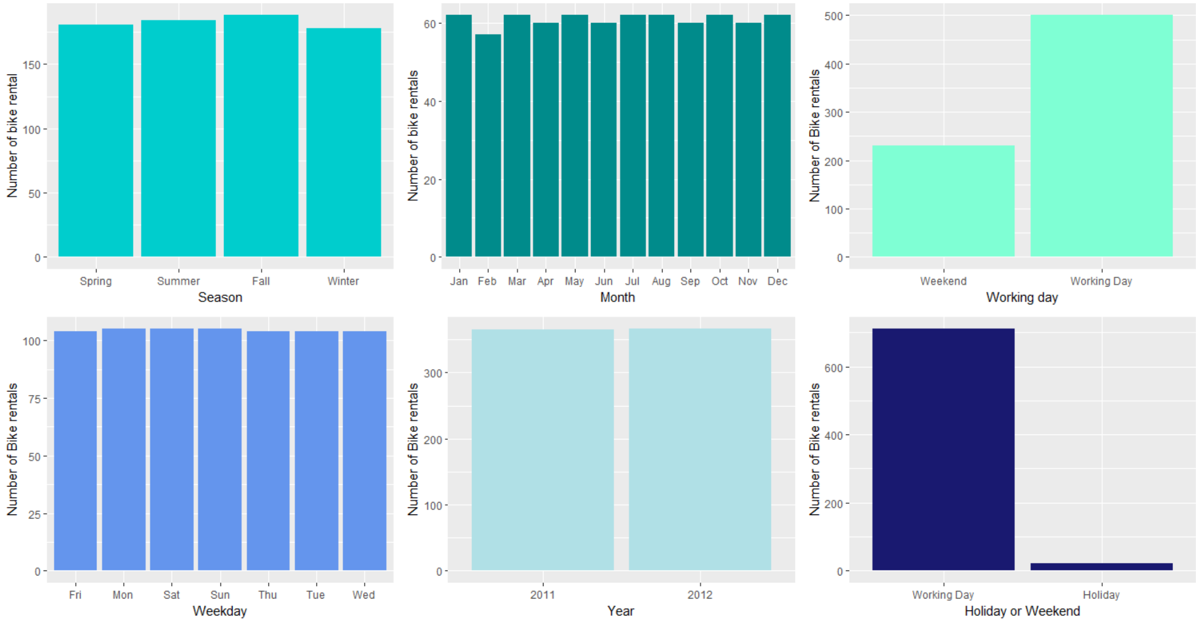

Fig. 2.1 : Categorical variables vs count

We can see that the distribution counts in season and find out that the number of bike rentals were higher in Fall. However, all the seasons had a good number of rentals. Month also did not affect the count of bike rentals. The rentals were almost same throughout the year. Weekends and Weekdays also performed the same way. This confirmed in the Weekday versus counts graph as we can see that Sunday and Saturday also had almost the same number of rental bookings as other working days. Also the year 2011 and 2012 had almost the same number of rentals. There was no significant increase from year to year.

2.4 Distribution of Continuous Variables

In the below graph we can see the distribution of continuous variables such as temperature, humidity, weathersit and windspeed.

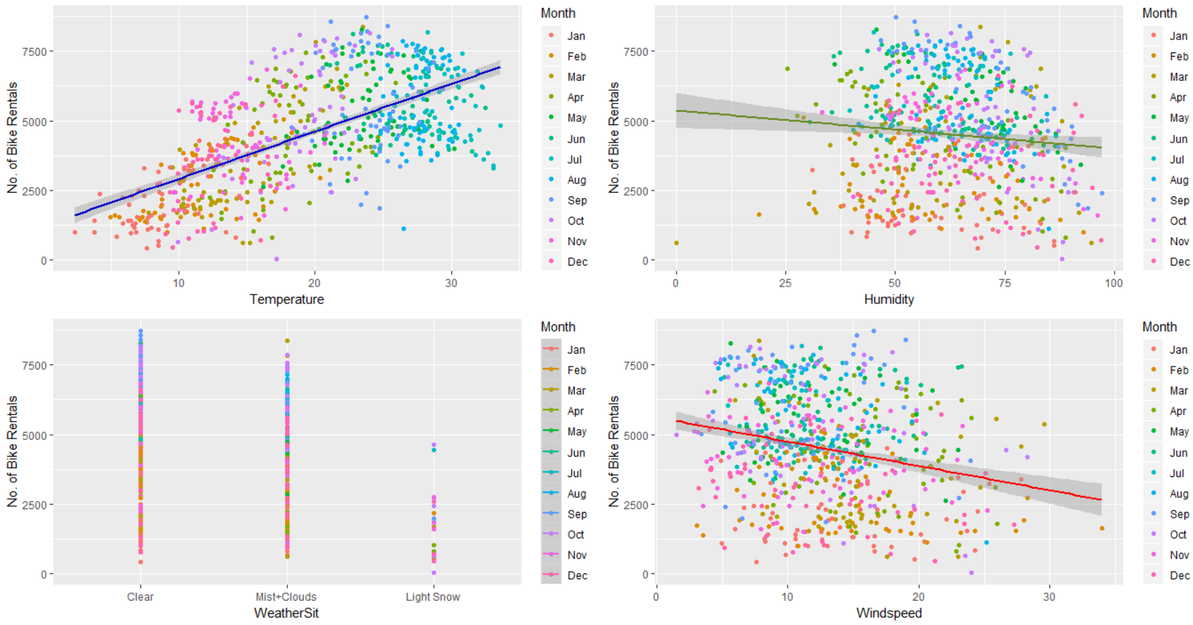

Fig. 2.2 : Categorical variables vs count

In the Temperature chart we can see that the number of rentals were low when the temperature was low(starting of the year) compared to the number of counts when the temperature was more(towards mid yeat). Humidity did not have a significant affect on the rentals count. However, Weathersit indicated that the when the weather was clear or not bad, there were good number of rentals compared to bad weather where we don't see much of the bookings.

2.5 Feature Selection

As we looked the dependency of dependent variable on other variable, we could see what are the variables which are affecting the number of counts. It is important for us to select the variables which are highly related to the dependent variable and reduce any error that we may have. This can be done using the Correlation plot or Correlograms.

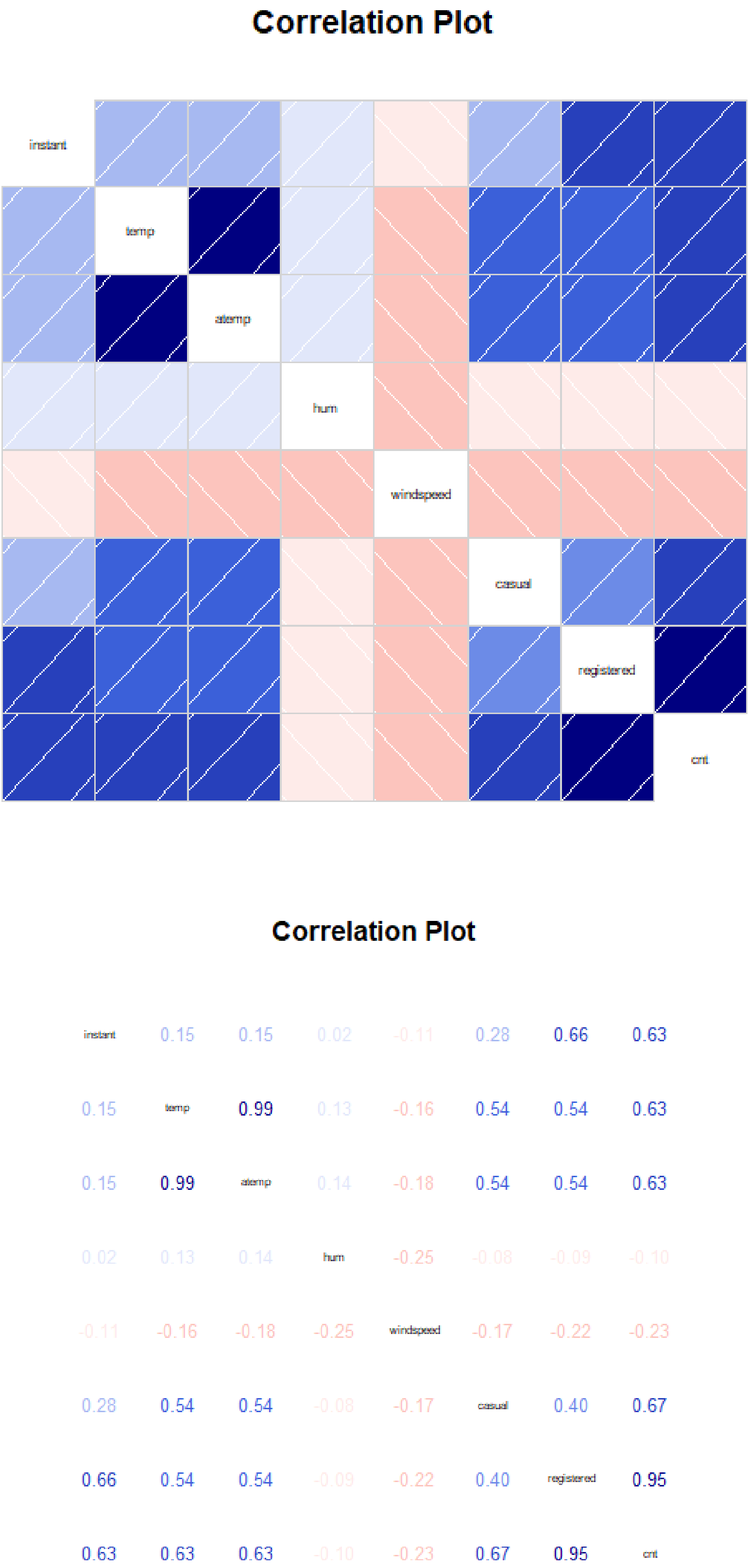

Fig. 2.3 : Correlation plot

Chapter 3

Modelling

3.1 Model Selection

Model Selection is where we select the suitable modelling technique based on the type of dependent variable. Since the number of bike rentals is a continuous variable we are going with Linear Regression and Random Forest.

3.2 Linear Regression

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

We can see that the Adjusted R squared is 85.6% which means we cans explain 85.6% of the data using our model. F-statistic is 483 and p-value is 2.2e-16 which can reject the null hypothesis that target variable does not depend on any of the predictor variables.

The MAPE is 21.5% and F-statistic is 483. Hence the accuracy of the model is 78.5%. This means our model is good.

```
Residuals:
    Min      1Q  Median      3Q     Max
-3403.3  -331.6    57.7   462.0  2878.2

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)             1160.302    335.295   3.461 0.000587 ***
seasonSummer            1130.586    212.105   5.330 1.51e-07 ***
seasonFall              1168.262    241.918   4.829 1.84e-06 ***
seasonWinter            1779.893    205.755   8.651  < 2e-16 ***
yr2012                  2123.619     68.152  31.160  < 2e-16 ***
mnthFeb                  192.668    165.856   1.162 0.245948
mnthMar                  634.760    190.335   3.335 0.000919 ***
mnthApr                  477.989    289.623   1.650 0.099515 .
mnthMay                  691.612    309.393   2.235 0.025849 *
mnthJun                  464.963    322.348   1.442 0.149830
mnthJul                  -88.353    357.220  -0.247 0.804753
mnthAug                  358.115    341.787   1.048 0.295266
mnthSep                  903.012    299.034   3.020 0.002664 **
mnthOct                  588.618    278.998   2.110 0.035393 *
mnthNov                 -111.182    263.779  -0.421 0.673579
mnthDec                  -17.929    206.943  -0.087 0.930994
weekdayMon              -243.845    127.810  -1.908 0.056999 .
weekdaySat               718.005    233.629   3.073 0.002237 **
weekdaySun               121.432    234.039   0.519 0.604102
weekdayThu               -66.019    126.022  -0.524 0.600611
weekdayTue               -86.060    125.954  -0.683 0.494769
weekdayWed               -96.091    120.893  -0.795 0.427098
workingdayWorking Day    588.082    205.867   2.857 0.004467 **
weathersitMist+Clouds   -470.474     91.186  -5.159 3.62e-07 ***
weathersitLight Snow   -2336.316    221.510 -10.547  < 2e-16 ***
temp                     101.305     12.383   8.181 2.51e-15 ***
hum                      -13.019      3.401  -3.828 0.000146 ***
windspeed                -34.491      7.071  -4.878 1.46e-06 ***
 ---
 Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

 Residual standard error: 746.9 on 483 degrees of freedom
 Multiple R-squared:  0.8642,	Adjusted R-squared:  0.8566
 F-statistic: 113.9 on 27 and 483 DF,  p-value: < 2.2e-16
```

3.3 Random Forest

Apart from Regression, we shall use one Classification model for the predictions. Number of trees used in this case is 300. MAPE for the model is 16.97% and MAE for the model is 479. Hence the accuracy is 83.03%.

Chapter 4

Conclusion

4.1 Model Evaluation

We have 2 models for predicting the count of rentals. Now, we need to decide on which one to choose.

We can compare the models using any of the following criteria :

1. Predictive Performance

2. Interpretability

3. Computational Efficiency

For our case, criteria 2 and 3 does not hold good. Hence we go with Predictive Performance.

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

4.3 Mean Absolute Percentage Error(MAPE)

MAPE can be calculated by the below formula and MAPE for Linear Regression is 21.5% and For Random Forest it is 16.97%.

MAPE = ((Actual_value – Predicted Value)/Actual Value)*100

4.3 Model Selection

As we have seen that from the below table, Random Forest model has better Accuracy and less MAPE. So we can choose Random Forest

Model	**MAPE**	**Accuracy**
Linear Regression - Model1	21.50%	78.50%
Random Forest - Model 2	16.97%	83.03%

Table 4.1 Model Accuracy

Chapter 5

R- Code

```
#################################### BIKE RENTAL PROJECT
########################################


#Clean the environment

rm(list = ls())


                # LOAD LIBRARIES AND IMPORT DATASET

#Set working directory

setwd("C:/Users/shrid/Downloads/edWisor")




#Load libraries

library("readr")

library("dplyr")

library("plyr")

library("corrplot")

library("ggplot2")

library("randomForest")

library("ggExtra")

library("ggpubr")

library("corrgram")
```

```
library("rpart")

library("DMwR")

library("Metrics")


#Import dataset

day = read.csv(file = "day.csv", header = TRUE, sep = ",", na.strings = c("", " ", "NA"))


#Structure of the data set

str(day)


#Let's check for the Missing values

missing_values = sapply(day,function(x){  sum(is.na(x))  })

missing_values        # We don't see any missing values



                    # FEATURE ENGINEERING



# season: Season (1:spring, 2:summer, 3:fall, 4:winter)

day$season <- factor( x = day$season, levels = c(1,2,3,4), labels = c("Spring",
"Summer", "Fall", "Winter"))


# yr: Year (0: 2011, 1:2012)
```

```
day$yr <- factor(x = day$yr, levels = c(0,1), labels = c("2011", "2012"))

#mnth: Month (1 to 12)

day$mnth <- factor(x = day$mnth, levels = c(1:12), labels = c("Jan", "Feb", "Mar",
"Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))

#holiday: weather day is holiday or not (extracted fromHoliday Schedule)
day$holiday <- factor(x = day$holiday,levels = c(0,1),labels = c("Working Day",
"Holiday") )

#weekday: Day of the week
day$weekday <- factor(x = day$weekday, levels = c(0:6), labels = c("Sun", "Mon",
"Tue", "Wed", "Thu", "Fri", "Sat" ))

#workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
day$workingday <- factor(x = day$workingday, levels = c(0,1), labels =
c("Weekend","Working Day"))

#weathersit: (extracted fromFreemeteo)
#1: Clear, Few clouds, Partly cloudy, Partly cloudy
#2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
```

```
#3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain +
Scattered clouds
#4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
day$weathersit <- factor( x = day$weathersit, levels = c(1:4), labels = c("Clear",
"Mist+Clouds", "Light Snow", "Heavy Rain"))


#temp: Normalized temperature in Celsius. The values are derived via
# temp = (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)
#t = temp*(39+8) -8 which is approximately t = temp *39
day$temp = day$temp*39


#atemp: Normalized feeling temperature in Celsius. The values are derived via
#(t-t_min)/(t_maxt_min), t_min=-16, t_max=+50 (only in hourly scale)
day$atemp = day$atemp*50


#hum: Normalized humidity. The values are divided to 100 (max)
day$hum = day$hum*100
```

#windspeed: Normalized wind speed. The values are divided to 67 (max)

day$windspeed = day$windspeed*67

#Let's change the variables to proper data types

day$dteday <- as.character(day$dteday)

day$mnth <- as.factor(day$mnth)

day$weekday = as.factor(as.character(day$weekday))

day$workingday = as.factor(as.character(day$workingday))

 # EXPLORATORY DATA ANALYSIS

#Let's see how different scenarios affect the Bike rentals

a = ggplot(day, aes(season)) + geom_bar(fill = "cyan3") + labs(x = "Season", y = "Number of bike rental")

#a

```
b = ggplot(day, aes(mnth)) + geom_bar(fill = "darkcyan") + labs(x = "Month", y =
("Number of bike rentals"))

#b


c = ggplot(day, aes(workingday)) + geom_bar(fill = "aquamarine") + labs(x =
"Working day", y="Number of Bike rentals")

#c


d = ggplot(day, aes(weekday)) + geom_bar(fill = "cornflowerblue") + labs(x =
"Weekday", y = "Number of Bike rentals")

#d


e = ggplot(day, aes(yr)) + geom_bar(fill = "powderblue")  + labs(x = "Year", y =
"Number of Bike rentals")

#e


f = ggplot(day, aes(holiday)) + geom_bar(fill = "midnightblue") + labs(x="Holiday or
Weekend", y = "Number of Bike rentals")

#f

ggarrange(a,b,c,d,e,f, widths = c(1,1))


#Let's see how different weather conditions affect the count
```

```
chart1 = ggplot(day, aes(x = temp, y = cnt, col = factor(mnth))) +
  geom_point() + labs( x = "Temperature", y = "No. of Bike Rentals", col = "Month") +
    stat_smooth(method = lm, col = "mediumblue")
#chart1


chart2 = ggplot(day, aes(x = hum, y = cnt, col = factor(mnth))) + geom_point() +
  labs( x = "Humidity", y = "No. of Bike Rentals", col = "Month") +
  stat_smooth(method = "lm", col = "olivedrab4")
#chart2


chart3 = ggplot(day, aes(x = weathersit, y=cnt, col = factor(mnth) )) + geom_point()
+ geom_smooth()+
  labs(x = "WeatherSit", y = "No. of Bike Rentals", col = "Month")
#chart3


chart4 = ggplot(day, aes(x = windspeed, y = cnt, col = factor(mnth))) + geom_point()
+
  labs(x = "Windspeed", y= "No. of Bike Rentals", col = "Month") +
stat_smooth(method = lm, col = "red")
#chart4


ggarrange(chart1, chart2, chart3, chart4, widths = c(1,1))
# Let's use the Correlation Plot to check the Co-efficient of Correlation and select correlated variables
```

```
p1 <- corrgram(day,order =  FALSE, main = "Correlation Plot")

p2 <-corrgram(day,order =  FALSE, main = "Correlation Plot", panel = panel.cor)


#Let's remove variables which do not help us in the prediction


data1 <- subset(day, select =
c(season,yr,mnth,weekday,workingday,weathersit,temp,hum,windspeed,cnt))

data1

set.seed(300)

index = sample(1:nrow(data1), as.integer(0.7*nrow(data1)))

train = data1[index,]

test = data1[-index,]


#Linear Regression


model1 = lm(formula = cnt~., data = train)

summary(model1)

prediction1 = predict(model1, test[,-10])


df = data.frame("actual" = test[,10], "pred" = prediction1)

head(df)
```

regr.eval(trues = test[,10], preds = prediction1, stats = c("mae", "mse", "rmse", "mape"))

require(stats)

mape(test[,10], prediction1)

 # Random Forest

model2 = randomForest(cnt~., data = train, tree = 100)

summary(model2)

prediction2 = predict(model2, test[,-10])

df = cbind(df, prediction2)

head(df)

regr.eval(trues = test[,10], preds = prediction2, stats = c("mae","mse","rmse","mape"))

mape(test[,10], prediction2)

```
data_model1 = pd.DataFrame({'actual': test.iloc[:,9], 'pred': prediction1})
data_model1.head()
```

	actual	pred
226	4338	4664.502080
181	5362	5411.791388
509	6770	6705.598181
606	7697	7181.200067
304	4068	3843.592190

Fig. 5.1 : Model 1 : Actual vs Predicted value

Chapter 6

Python Code

```
import pandas as pd

from ggplot import *

import numpy as np

import os

import matplotlib.pyplot as plt

import pylab

import statsmodels as sm

import seaborn as sn

#Import the data

os.chdir("C:¥¥Users¥shrid¥Downloads¥edWisor")


os.getcwd()


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


#Let's check the structure of data

day.shape

day.dtypes
```

#Let's check for Missing values

day.isnull().sum()

#We don't see any null values, So we are good to go

#FEATURE ENGINEERING

#season: Season (1:springer, 2:summer, 3:fall, 4:winter)

day["season"] = day["season"].replace([1,2,3,4],["Spring", "Summer", "Fall", "Winter"])

day["yr"] = day["yr"].replace([0,1],["2011", "2012"])

 # yr: Year (0: 2011, 1:2012)

day["mnth"] = day["mnth"].replace([1,2,3,4,5,6,7,8,9,10,11,12], ["Jan", "Feb", "mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

#holiday: weather day is holiday or not (extracted fromHoliday Schedule)

day["holiday"] = day["holiday"].replace([0,1],["Working Day", "Holiday"])

#weekday: Day of the week

day["weekday"] = day["weekday"].replace([0,1,2,3,4,5,6],["Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"])

#workingday: If day is neither weekend nor holiday is 1, otherwise is 0.

```
day["workingday"] = day["workingday"].replace([0,1],["Weekoff","Weekday"])

#weathersit: (extracted fromFreemeteo)
#1: Clear, Few clouds, Partly cloudy, Partly cloudy
#2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
#3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
#4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
day["weathersit"] = day["weathersit"].replace([1,2,3,4],["Clear","Misty+Clouds","Light Snow","Heavy Rain"])

#temp: Normalized temperature in Celsius. The values are derived via
# temp = (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)
#t = temp*(39+8) -8 which is approximately t = temp *39
day["temp"] = day["temp"]*39

#atemp: Normalized feeling temperature in Celsius. The values are derived via
#(t-t_min)/(t_maxt_min), t_min=-16, t_max=+50 (only in hourly scale)

day["atemp"] = day["atemp"]*50

#hum: Normalized humidity. The values are divided to 100 (max)
day["hum"] = day["hum"]*100
```

```
#windspeed: Normalized wind speed. The values are divided to 67 (max)
day["windspeed"] = day["windspeed"]*67

day.head()

a = ggplot(day, aes("season")) + geom_bar(fill = "cyan") + labs(x = "Season", y = 
"Number of bike rental")
a

b = ggplot(day, aes("mnth")) + geom_bar(fill = "darkcyan") + labs(x = "Month", y = 
("Number of bike rentals"))
b

c = ggplot(day, aes("workingday")) + geom_bar(fill = "aquamarine") + labs(x = 
"Working day", y="Number of Bike rentals")
c

d = ggplot(day, aes("weekday")) + geom_bar(fill = "cornflowerblue") + labs(x = 
"Weekday", y = "Number of Bike rentals")
d

e = ggplot(day, aes("yr")) + geom_bar(fill = "powderblue")  + labs(x = "Year", y = 
"Number of Bike rentals")
```

```
e

f = ggplot(day, aes("holiday")) + geom_bar(fill = "midnightblue") + labs(x="Holiday or Weekend", y = "Number of Bike rentals")

f

chart1 = ggplot(day, aes(x = "temp", y = "cnt", color = "mnth")) + geom_point() + labs( x = "Temperature", y = "No. of Bike Rentals")

chart1

chart2 = ggplot(day, aes(x = "hum", y = "cnt", color = "mnth")) + geom_point() + labs( x = "Humidity", y = "No. of Bike Rentals")

chart2

chart3 = ggplot(day, aes(x = "weathersit", y="cnt", color = "mnth")) + geom_point() + labs(x = "WeatherSit", y = "No. of Bike Rentals")

chart3

chart4 = ggplot(day, aes(x = "windspeed", y = "cnt", color = "mnth")) + geom_point() + labs(x = "Windspeed", y= "No. of Bike Rentals")

chart4

sn.heatmap(day.cov().corr())

plt.show()
```

```
day1 = pd.read_csv("day.csv")

data1 = day1.drop(columns =
["instant","dteday","holiday","registered","casual","atemp"])

train,test = train_test_split(data1, test_size = 0.3, random_state = 300)


#linear Regression


from sklearn.model_selection import train_test_split

import statsmodels.api as sm

from sklearn.metrics import mean_squared_error


#Let's Train the model

model1 = sm.OLS(train.iloc[:,9].astype(float), train.iloc[:,0:9].astype(float)).fit()

model1.summary()


#Let's predict the Summary of Test data

prediction1 = model1.predict(test.iloc[:,0:9])


data_model1 = pd.DataFrame({'actual': test.iloc[:,9], 'pred': prediction1})

data_model1.head()


#Let's calculate MAPE

def MAPE(y_actual,y_pred):
```

```
    mape = np.mean(np.abs((y_actual - y_pred)/y_actual)*100)
    return mape
MAPE(test.iloc[:,9],prediction1)


#Random Forest
from sklearn.ensemble import RandomForestRegressor


model2 = 
RandomForestRegressor(n_estimators=100,random_state=300).fit(train.iloc[:,0:9],
train.iloc[:,9])


prediction2 = model2.predict(test.iloc[:,0:9])


data_model2 = pd.DataFrame({"actual" : test.iloc[0:,9],"pred" : prediction2})


MAPE(test.iloc[:,9], prediction2)
```

```
data_model2 = pd.DataFrame({"actual" : test.iloc[0:,9],"pred" : prediction2})
```

```
data_model2
```

	actual	pred
226	4338	4709.62
181	5362	4977.20
509	6770	6570.64
606	7697	7592.50
304	4068	3759.79

Fig. 6.1 : Model 2 : Actual vs Predicted value