DATASCIENCE WITH R

PROJECT 1

2021

PROJECT WRITE UP
WALMART RETAIL ANALYSIS

Reeta Singh

WALMART RETAIL ANALYSIS

7/18/2021

WALMART RETAIL ANALYSIS

REETA SINGH

OBJECTIVE:-

A detailed analysis of the stores of Walmart.

BUSINESS SCENARIO:

DESCRIPTION

One of the leading retail stores in the US, Walmart, would like to predict the sales and demand accurately. There are certain events and holidays which impact sales on each day. There are sales data available for 45 stores of Walmart. The business is facing a challenge due to unforeseen demands and runs out of stock sometimes, due to the inappropriate machine learning algorithm.

An ideal ML algorithm will predict demand accurately and ingest factors like economic conditions including CPI, Unemployment Index, etc.

Walmart runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of all, which are the Super Bowl, Labour Day, Thanksgiving, and Christmas. The weeks including these holidays are weighted five times higher in the evaluation than non-holiday weeks. Part of the challenge presented by this competition is modeling the effects of markdowns on these holiday weeks in the absence of complete/ideal historical data. Historical sales data for 45 Walmart stores located in different regions are available.

DATASET VARIABLES: -

This is the historical data which covers sales from 2010-02-05 to 2012-11-01, in the file Walmart_Store_sales. Within this file are the following fields:

- Store - the store number
- Date - the week of sales
- Weekly_Sales - sales for the given store
- Holiday_Flag - whether the week is a special holiday week 1 – Holiday week 0 – non-holiday week
- Temperature - Temperature on the day of sale
- Fuel_Price - Cost of fuel in the region
- CPI – Prevailing consumer price index

- Unemployment - Prevailing unemployment rate

Holiday Events

Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13
Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13
Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13
Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

EXPECTATION/GOALS :-

Analysis Tasks

Basic Statistics tasks

- Which store has maximum sales?
- Which store has maximum standard deviation i.e., the sales vary a lot. Also, find out the coefficient of mean to standard deviation?
- Which store/s has good quarterly growth rate in Q3'2012
- Some holidays have a negative impact on sales. Find out holidays which have higher sales than the mean sales in non-holiday season for all stores together?
- Provide a monthly and semester view of sales in units and give insights?

Statistical Model

For Store 1 – Build prediction models to forecast demand.

- Linear Regression – Utilize variables like date and restructure dates as 1 for 5 Feb 2010 (starting from the earliest date in order). Hypothesize if CPI, unemployment, and fuel price have any impact on sales.
- Change dates into days by creating new variable.

Select the model which gives best accuracy.

ANALYSIS INFORMATION :-

Basic Steps -

1. Import the Walmart Retail dataset.
2. Load the required libraries.
3. View the structure, dimension of dataset.
4. Check for missing values.

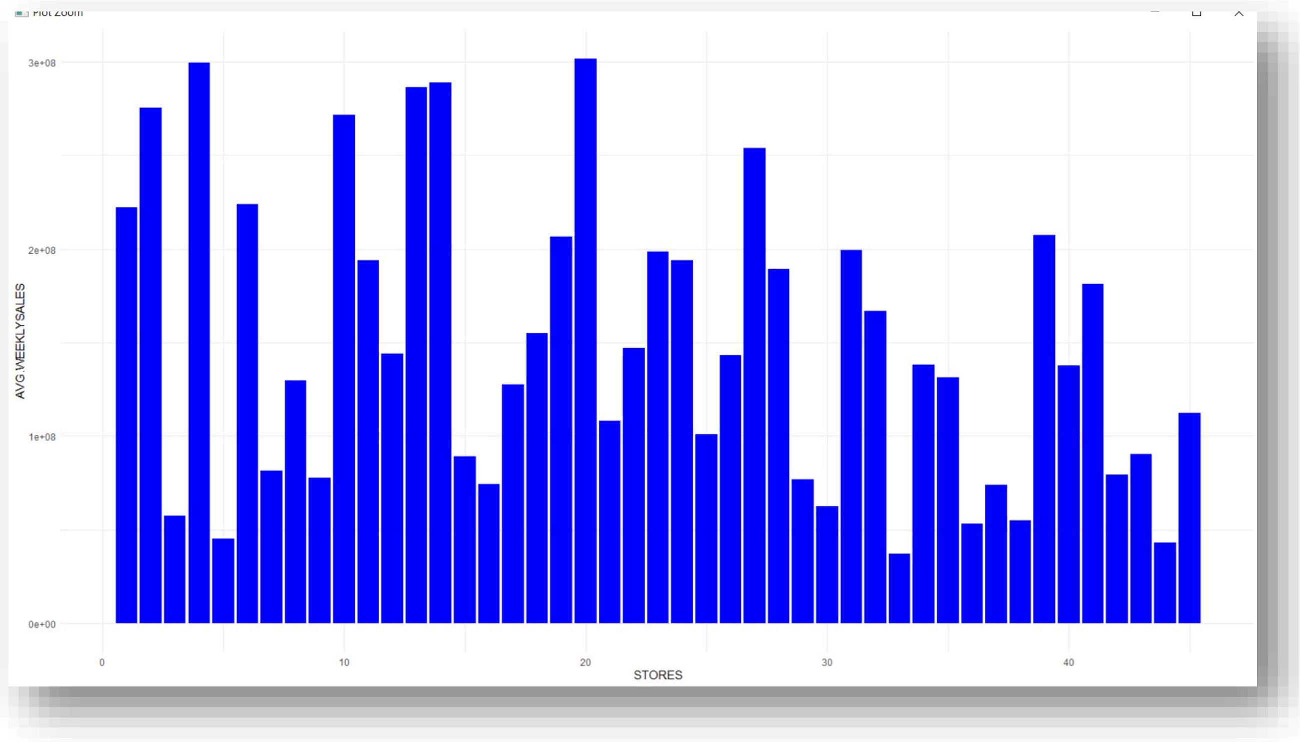
```r
library(ggplot2) # for visualization
library(dplyr)
library(lubridate)
library(tidyverse)
library(car)
library(corrplot) # for corrplot
library(caTools) # for splitting dataset
library(MLmetrics)


setwd("~/SIMPLILEARN/PROJECTS/R PROJECT/1572585064_walmart_store_sales")

#---------------------------------LOADING DATA-------------------------------------
WALMART_DATA <-  read.csv('Walmart_Store_sales.csv',stringsAsFactors = F)

WALMART_DATA$Date <- dmy(WALMART_DATA$Date)

View(WALMART_DATA)

str(WALMART_DATA)

dim(WALMART_DATA)

#--------------------------CHECKING FOR MISSING VALUES------------------------------------------

any(is.na(WALMART_DATA))
```

Task 1 –

Finding the Store with maximum sales-

Aggregate sum of Walmart Weekly Sales by Stores. Select the maximum sales from aggregated data.

```r
#--------------------------FINDING THE STORE WITH MAXIMUM SALES -------------------------------

STORE_MAX_SALE <- aggregate(WALMART_DATA$Weekly_Sales, by = list(WALMART_DATA$Store), sum , na.rm = T)
STORE_MAX_SALE <- STORE_MAX_SALE %>% rename(STORES = Group.1, AVG.WEEKLYSALES = x)

STORE_MAX_SALE = STORE_MAX_SALE[which.max(unlist(STORE_MAX_SALE$x)),]
STORE_MAX_SALE
str(STORE_MAX_SALE)

PLT_1 <- ggplot(data=STORE_MAX_SALE, aes(x=STORES, y=AVG.WEEKLYSALES)) +
      geom_bar(stat="identity" , fill="blue",colors ='YELLOW') + theme_minimal()

PLT_1


# STORE 20 HAS MAXIMUM SALES - $301397792
```



Task 2 –

To find out which store has maximum standard deviation i.e., the sales vary a lot. Also, find out the coefficient of mean to standard deviation.

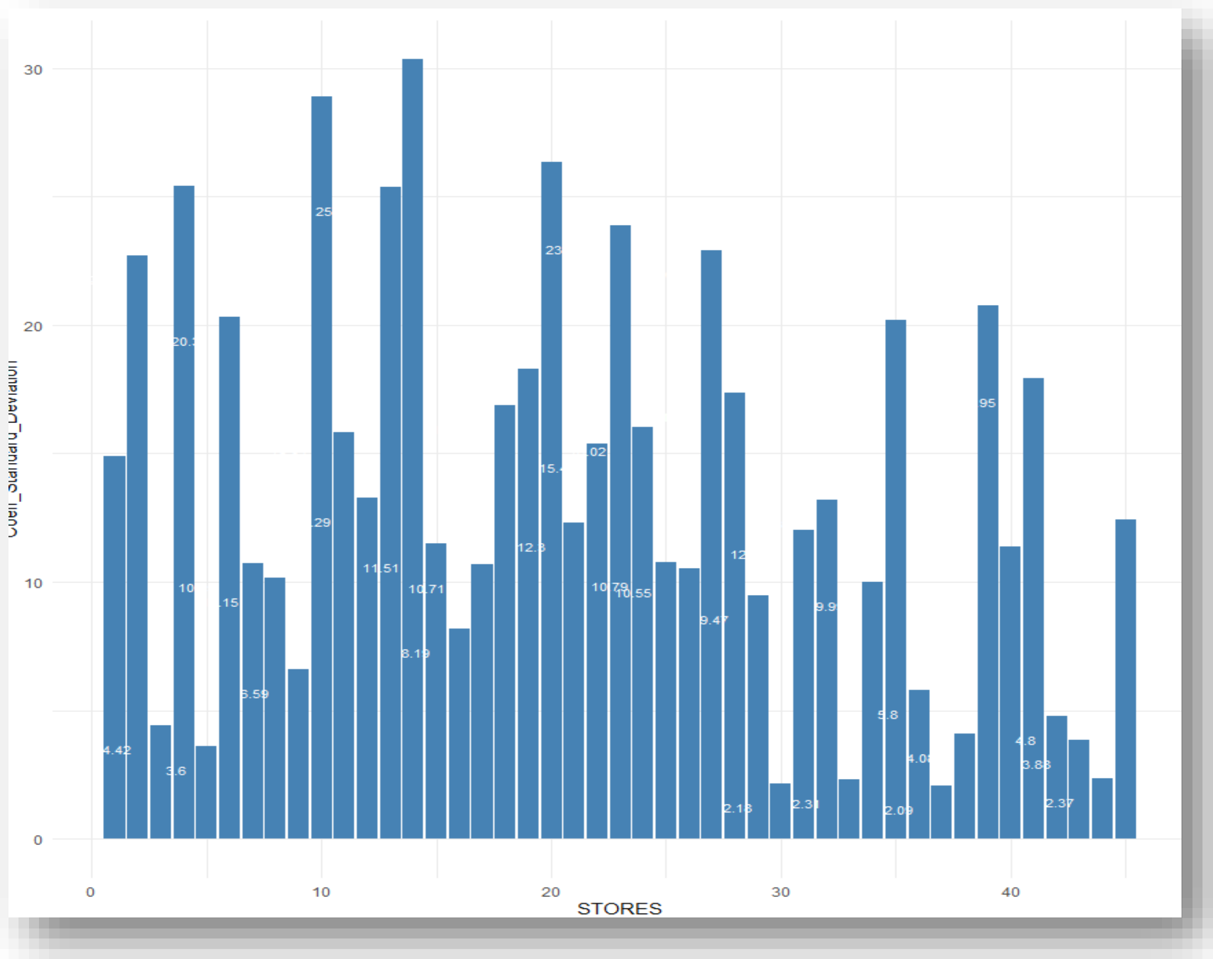
Aggregated sum of Walmart Weekly Sales by Stores. Applied Standard deviation and select the maximum value from the result.

To find the coefficient of mean of standard deviation used the following formula-

Coefficient of Mean of Standard deviation (CV) =

Standard Deviation/ mean(WALMART_Weekly_Sales) * 100

```r
#-------------------------FINDING STANDARD DEVIATION --------------------------------

STD <- aggregate(WALMART_DATA$Weekly_Sales, by = list(WALMART_DATA$Store), sd)
STD <- STD %>% rename(STORES = Group.1, STANDARD_DEVIATION = x)
STD = STD[which.max(unlist(STD$STANDARD_DEVIATION)),]
STD

PLT_1 <- ggplot(data=STD, aes(x=STORES, y=STANDARD_DEVIATION)) +
  geom_bar(stat="identity" , fill="steelblue",colors ='black') + theme_minimal() + coord_flip()
+ geom_text(aes(label=y),size = 3, hjust = 2, vjust = 2,color = 'white')


PLT_1

# STORE 14 HAS MAXIMUM STANDARD DEVIATION - $317569.9

#----------------------------COEFFICIENT OF MEAN OF STD--------------------
Coeff_Standard_Deviation <- STD$STANDARD_DEVIATION / mean(WALMART_DATA$Weekly_Sales) * 100
STD <- mutate(STD,Coeff_Standard_Deviation)
View(STD)
Coeff_Standard_Deviation


PLT_1 <- ggplot(data=STD, aes(x=STORES, y=Coeff_Standard_Deviation)) +
  geom_bar(stat="identity",fill = 'steelblue',colors = 'black') + theme_minimal()

PLT_1 <- PLT_1 + geom_text(aes(label=round(Coeff_Standard_Deviation,digits = 2)),size = 3,hjust = 2, vjust = 3,color = 'white')

PLT_1

#COEFFICIENT OF MEAN OF STANDARD DEVIATION OF STORE 14 = 30.33244
```



Task 3 : -

Which store/s has good quarterly growth rate in Q3'2012

To perform this task first we divide the date into Quarters and add it to the table. To find Q3 2012 growth rate we filter Q2 and Q3 data for year 2012 and do an aggregate sum on Weekly Sales.

Then we apply the formula

Growth Rate = (Q3 – Q2/ Q2) * 100

Sort the Growth Rate descending and select the top row.

```r
#-------------------------QUARTERLY GROWTH RATE ------------------------------


QUARTERS <- quarters(WALMART_DATA$Date)
WALMART_DATA = mutate(WALMART_DATA,QUARTERS)

WALMART_DATA <- mutate(WALMART_DATA,Year = as.numeric(format(WALMART_DATA$Date, "%Y")))

Q2 <- filter(WALMART_DATA, QUARTERS == 'Q2' & Year == '2012')
Q3 <- filter(WALMART_DATA, QUARTERS == 'Q3' & Year == '2012')

Q2 <- aggregate(Q2$Weekly_Sales,by= list(Store = Q2$Store),sum)
Q3 <- aggregate(Q3$Weekly_Sales,by= list(Store = Q3$Store),sum)

str(Q2)

temp <- merge(Q2,Q3,by = "Store")

temp <- temp %>% rename(Q2 = x.x, Q3 = x.y)

temp['Growth_Ratio'] <- NA

for(i in 1:nrow(temp)){ temp$Growth_Ratio <- ((temp$Q3 - temp$Q2) / temp$Q2) * 100}


temp <- temp[order(temp$Growth_Ratio, decreasing = TRUE), ]

head(temp , n=5)


#------------STORE 7 HAS HIGHEST QUARTERLY GROWTH RATE IN Q32012-----------------------------
#
```

Task 4 :-

Some holidays have a negative impact on sales. Find out holidays which have higher sales than the mean sales in non-holiday season for all stores together .

To find holiday impact on sales first we divide the data into Holiday and non holiday flag. For Non holiday take the mean of the sales.

For holiday's data ,categorize data into 4 holidays i.e Super Bowl, Labour Day , Thanksgiving and Christmas Sales. Then take out the mean of each sale and compare the values.

The output is that Thanksgiving has higher sales than any other sales day or non holiday sales.

```r
#-------------------------HOLIDAYS WHICH HAVE HIGHER SALES-----------------------------------

NON_HOL_FLAG <- filter(WALMART_DATA[1:4],Holiday_Flag == 0)
Mean_non_hol_flag <- mean(NON_HOL_FLAG$x)  #----------Mean of all stores on non holiday - 1046965--------------------------


WALMART_DATA$New_holiday <- NA
str(WALMART_DATA)


#Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13
#Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13
#Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13
#Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

Super <- c('2011-02-11','2010-02-12','2012-02-10' ,'2013-02-08')
Labour <- c('2010-09-10','2011-09-09','2012-09-07','2013-09-06')
ThGiving <- c('2010-11-26','2011-11-25','2012-11-23','2013-11-29')
Cmas <- c('2010-12-31','2011-12-30','2012-12-28','2013-12-27')


WALMART_DATA$New_holiday <- as.character(WALMART_DATA$New_holiday)
WALMART_DATA$Date <- as.character(WALMART_DATA$Date)

for(i in 1: nrow(WALMART_DATA))
{
  if(WALMART_DATA$Holiday_Flag[i] == 1){
    print(WALMART_DATA$Date[i])
  if(WALMART_DATA$Date[i] %in% (Super))
  {WALMART_DATA$New_holiday[i] <- 'S'}
  else if(WALMART_DATA$Date[i] %in% (Labour))
  { WALMART_DATA$New_holiday[i] <- 'L'}
  else if(WALMART_DATA$Date[i] %in% (ThGiving))
  {WALMART_DATA$New_holiday[i] <-'T'}
  else if (WALMART_DATA$Date[i] %in% (Cmas))
  {WALMART_DATA$New_holiday[i] <- 'C'}
  else{WALMART_DATA$New_holiday[i] <- 'H'}
  }
}

View(WALMART_DATA)
```

```r
View(WALMART_DATA)


SuperBowl_Sales <- filter(WALMART_DATA, Holiday_Flag ==1, New_holiday == 'S')
MEAN_SUPERBOWL <- mean((SuperBowl_Sales$Weekly_Sales))
MEAN_SUPERBOWL

LABOUR_SALES <- filter(WALMART_DATA, Holiday_Flag ==1, New_holiday == 'L')
MEAN_LABOUR <- mean((LABOUR_SALES$Weekly_Sales))
MEAN_LABOUR


THNKSGIV_SALES <- filter(WALMART_DATA, Holiday_Flag ==1, New_holiday == 'T')
MEAN_THNKSGIV <- mean((THNKSGIV_SALES$Weekly_Sales))
MEAN_THNKSGIV


CHRISTMAS_SALES <- filter(WALMART_DATA, Holiday_Flag ==1, New_holiday == 'C')
MEAN_CHRISTMAS <- mean((CHRISTMAS_SALES$Weekly_Sales))
MEAN_CHRISTMAS

Mean_non_hol_flag


#-----------------------OUTPUT - THANKSGIVING SALES IS HIGHER THAN AVG NON HOLIDAY SALES-----------------------
```

Task 4 :-

Provide a monthly and semester view of sales in units and give insights

For this analysis I made 3 functions to extract Day,Month and Year and passed the Walmart Sales Data Date as parameter. Then extract the monthly and Semester data by grouping and summarizing the data

```
#-----------------------OUTPUT - THANKSGIVING SALES IS HIGHER THAN AVG NON HOLIDAY SALES----------------

#----------------Extracting month from Date and adding to dataframe-------------------------

df_date = ExtractMonth(WALMART_DATA$Date)
WALMART_DATA = cbind(WALMART_DATA,df_date)
names(WALMART_DATA)[names(WALMART_DATA)=="df_date"] <- "Month"


df_year = ExtractYear(WALMART_DATA$Date)
WALMART_DATA = cbind(WALMART_DATA,df_year)
names(WALMART_DATA)[names(WALMART_DATA)=="df_year"] <- "Year"

df_day = ExtractDay(WALMART_DATA$Date)
WALMART_DATA = cbind(WALMART_DATA,df_day)
names(WALMART_DATA)[names(WALMART_DATA)=="df_day"] <- "Day"


##ExtractDate(as.character('2/19/20 20:31'))


############----------FUNCTION TO EXTRACT MONTH------------------------------------------

ExtractMonth <- function(Date){

  ExtractedDate <- as.Date(Date, format="%m/%d/%Y")
  month <- format(ExtractedDate,"%b")
  return(month)

}
```

```
#---------------------FUNCTION TO EXTRACT YEAR-----------------------------------------
ExtractYear <- function(Date){

  ExtractedDate <- as.Date(Date, format="%m/%d/%Y")
  year <- year(ExtractedDate)
  return(year)

}

#---------------------FUNCTION TO EXTRACT DAY------------------------------------------

ExtractDay <- function(Date){

  ExtractedDate <- as.Date(Date, format="%m/%d/%Y")
  day <- day(ExtractedDate)
  return(day)

}

#------------------------------------------------------------------------------------------------
monthlyData <- WALMART_DATA %>% group_by(Month,Year) %>% summarise(Mean_Monthly =
                                                              mean(Weekly_Sales))
View(monthlyData)

Semester <- semester(WALMART_DATA$Date,with_year = T)
WALMART_DATA <- mutate(WALMART_DATA, Semester)

semester_data <- WALMART_DATA %>% group_by(Semester,Year) %>% summarise(Mean_Semester= mean(Weekly_Sales))
View(semester_data)
```

Statistical Model:-

Linear Regression – Utilize variables like date and restructure dates as 1 for 5 Feb 2010 (starting from the earliest date in order).

```
WALMART_DATA %>% arrange(mdy(WALMART_DATAE1$Date))
WALMART_DATA$RESTRUCT_DATE=row_number(WALMART_DATA$Date)
View(WALMART_DATA)
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	RESTRUCT_DATE
1	1	2010-02-05	1643690.9	0	42.31	2.572	211.0964	8.106	1
2	1	2010-02-12	1641957.4	1	38.51	2.548	211.2422	8.106	46
3	1	2010-02-19	1611968.2	0	39.93	2.514	211.2891	8.106	91
4	1	2010-02-26	1409727.6	0	46.63	2.561	211.3196	8.106	136
5	1	2010-03-05	1554806.7	0	46.50	2.625	211.3501	8.106	181
6	1	2010-03-12	1439541.6	0	57.79	2.667	211.3806	8.106	226
7	1	2010-03-19	1472515.8	0	54.58	2.720	211.2156	8.106	271
8	1	2010-03-26	1404429.9	0	51.45	2.732	211.0180	8.106	316
9	1	2010-04-02	1594968.3	0	62.27	2.719	210.8204	7.808	361
10	1	2010-04-09	1545418.5	0	65.86	2.770	210.6229	7.808	406
11	1	2010-04-16	1466058.3	0	66.32	2.808	210.4887	7.808	451
12	1	2010-04-23	1391256.1	0	64.84	2.795	210.4391	7.808	496
13	1	2010-04-30	1425100.7	0	67.41	2.780	210.3895	7.808	541
14	1	2010-05-07	1603955.1	0	72.55	2.835	210.3400	7.808	586
15	1	2010-05-14	1494251.5	0	74.78	2.854	210.3374	7.808	631

Task:

Change dates into days by creating new variable.

```
#Change dates into days by creating new variable
WALMART_DATA <- WALMART_DATA%>%mutate(Days=day(Date))
View(WALMART_DATA)
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	RESTRUCT_DATE	Days
1	1	2010-02-05	1643690.9	0	42.31	2.572	211.0964	8.106	1	5
2	1	2010-02-12	1641957.4	1	38.51	2.548	211.2422	8.106	46	12
3	1	2010-02-19	1611968.2	0	39.93	2.514	211.2891	8.106	91	19
4	1	2010-02-26	1409727.6	0	46.63	2.561	211.3196	8.106	136	26
5	1	2010-03-05	1554806.7	0	46.50	2.625	211.3501	8.106	181	5
6	1	2010-03-12	1439541.6	0	57.79	2.667	211.3806	8.106	226	12
7	1	2010-03-19	1472515.8	0	54.58	2.720	211.2156	8.106	271	19
8	1	2010-03-26	1404429.9	0	51.45	2.732	211.0180	8.106	316	26
9	1	2010-04-02	1594968.3	0	62.27	2.719	210.8204	7.808	361	2
10	1	2010-04-09	1545418.5	0	65.86	2.770	210.6229	7.808	406	9
11	1	2010-04-16	1466058.3	0	66.32	2.808	210.4887	7.808	451	16
12	1	2010-04-23	1391256.1	0	64.84	2.795	210.4391	7.808	496	23
13	1	2010-04-30	1425100.7	0	67.41	2.780	210.3895	7.808	541	30
14	1	2010-05-07	1603955.1	0	72.55	2.835	210.3400	7.808	586	7
15	1	2010-05-14	1494251.5	0	74.78	2.854	210.3374	7.808	631	14

Task: -

For Store 1 – Build prediction models to forecast demand.

Hypothesize if CPI, unemployment, and fuel price have any impact on sales.

PREPROCESSING STEPS -

1. First step is to filter data for Store1 from Walmart dataset to perform further analysis.
2. Drop unnecessary columns 'Store','Date','RESTRUCT_DATE','Days'.

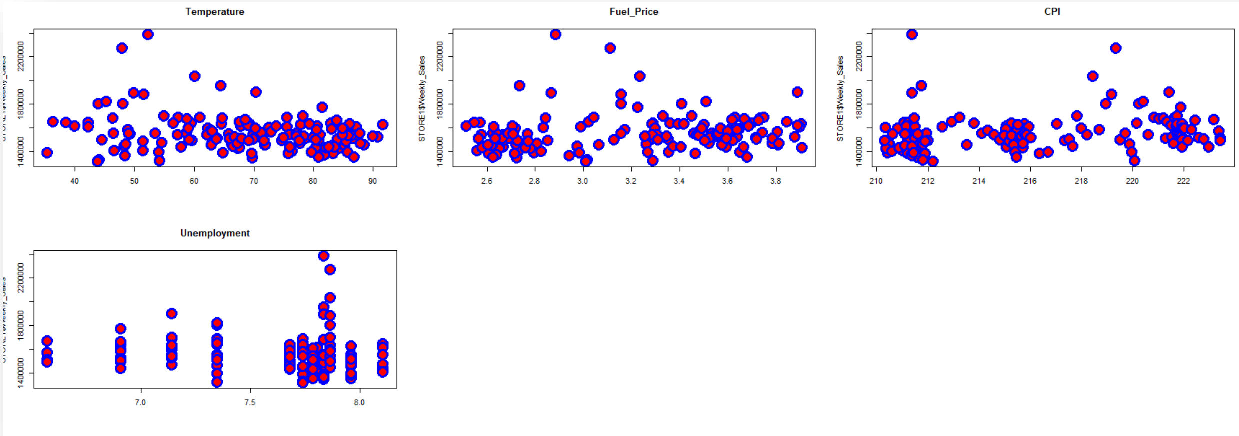
```
STORE1 <- filter(WALMART_DATA,Store == 1)
STORE1
dim(STORE1)
str(STORE1)
View(STORE1)
summary(STORE1)


dropc <- c('Store','Date','RESTRUCT_DATE','Days')

STORE1 <- STORE1[,!names(STORE1) %in% dropc]
```

	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
1	1643691	0	42.31	2.572	211.0964	8.106
2	1641957	1	38.51	2.548	211.2422	8.106
3	1611968	0	39.93	2.514	211.2891	8.106
4	1409728	0	46.63	2.561	211.3196	8.106
5	1554807	0	46.50	2.625	211.3501	8.106
6	1439542	0	57.79	2.667	211.3806	8.106
7	1472516	0	54.58	2.720	211.2156	8.106
8	1404430	0	51.45	2.732	211.0180	8.106
9	1594968	0	62.27	2.719	210.8204	7.808
10	1545419	0	65.86	2.770	210.6229	7.808
11	1466058	0	66.32	2.808	210.4887	7.808
12	1391256	0	64.84	2.795	210.4391	7.808
13	1425101	0	67.41	2.780	210.3895	7.808
14	1603955	0	72.55	2.835	210.3400	7.808
15	1494252	0	74.78	2.854	210.3374	7.808

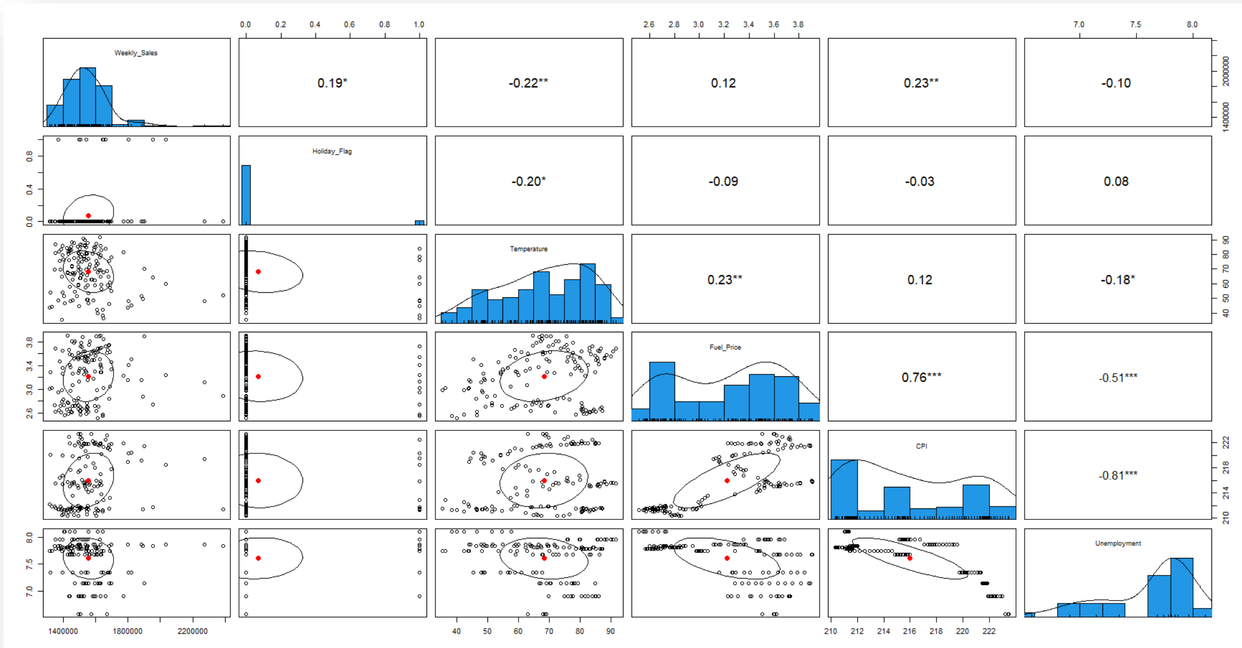
---

3. Next step is to find the correlation between the dependent and independent columns

```
#--------------------------FIND CORRELATION BETWEEN FEATURES---------------------------------------

cor_temp <- cor(STORE1$Weekly_Sales, STORE1$Temperature)  #----------- -0.2227006
cor_fuel <- cor(STORE1$Weekly_Sales, STORE1$Fuel_Price)   #-----------  0.1245916
cor_cpi <- cor(STORE1$Weekly_Sales, STORE1$CPI)          #-----------  0.2254077
cor_unemployment <- cor(STORE1$Weekly_Sales, STORE1$Unemployment) #----------- -0.09795539
cor_holidayflag <- cor(STORE1$Weekly_Sales, STORE1$Holiday_Flag) #-----------  0.1949052

par(mfrow=c(3,3))
for(i in 3:6){
  plot(STORE1[,i],
       STORE1$Weekly_Sales,
       main=names(STORE1[i]),
       ylab=names(STORE1$Weekly_Sales),
       xlab="", col='indianred4')
}
```

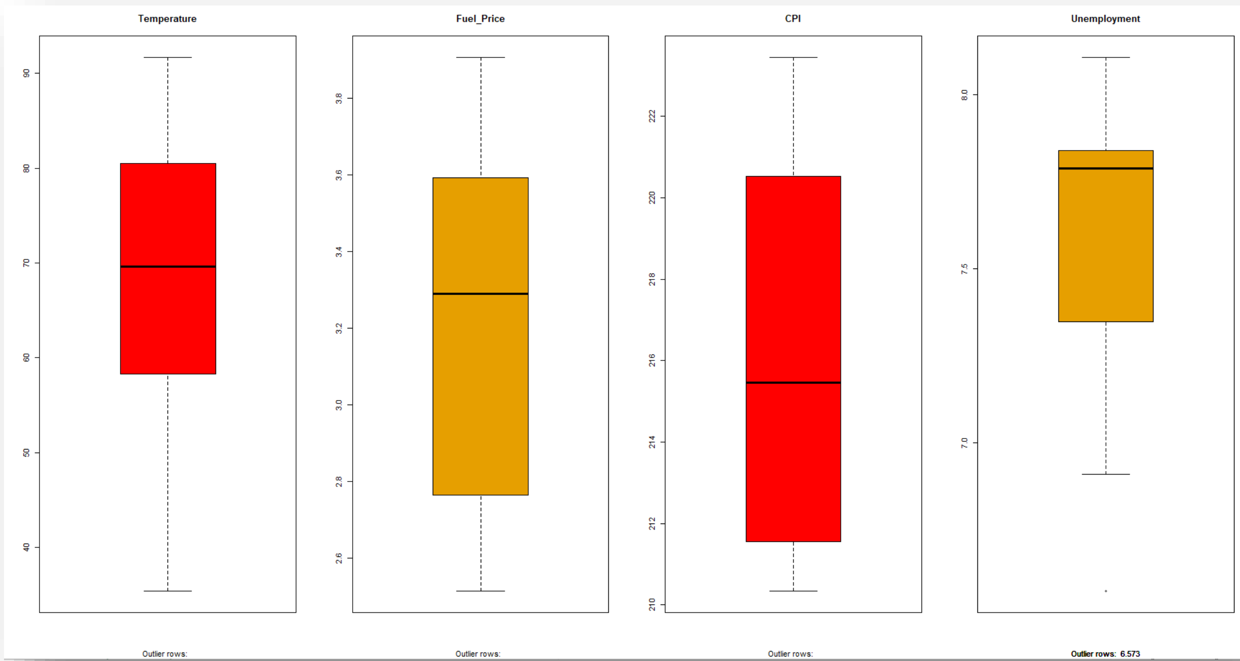



```r
pairs.panels(STORE1,
             smooth = F,      # If TRUE, draws loess smooths
             scale = FALSE,      # If TRUE, scales the correlation text font
             density = TRUE,     # If TRUE, adds density plots and histograms
             ellipses = TRUE,    # If TRUE, draws ellipses
             method = "pearson", # Correlation method (also "spearman" or "kendall")
             pch = 21,           # pch symbol
             lm = FALSE,         # If TRUE, plots linear fit rather than the LOESS (smoothed) fit
             cor = TRUE,         # If TRUE, reports correlations
             jiggle = FALSE,     # If TRUE, data points are jittered
             factor = 2,         # Jittering factor
             hist.col = 4,       # Histograms color
             stars = TRUE,       # If TRUE, adds significance level with stars
             ci = TRUE)

#------------From the correlation score we see that Temperature,Unemployment are negatively correlated and
#               Fuel,CPI and Holidays have positive impact on sales.------------------------------------
```



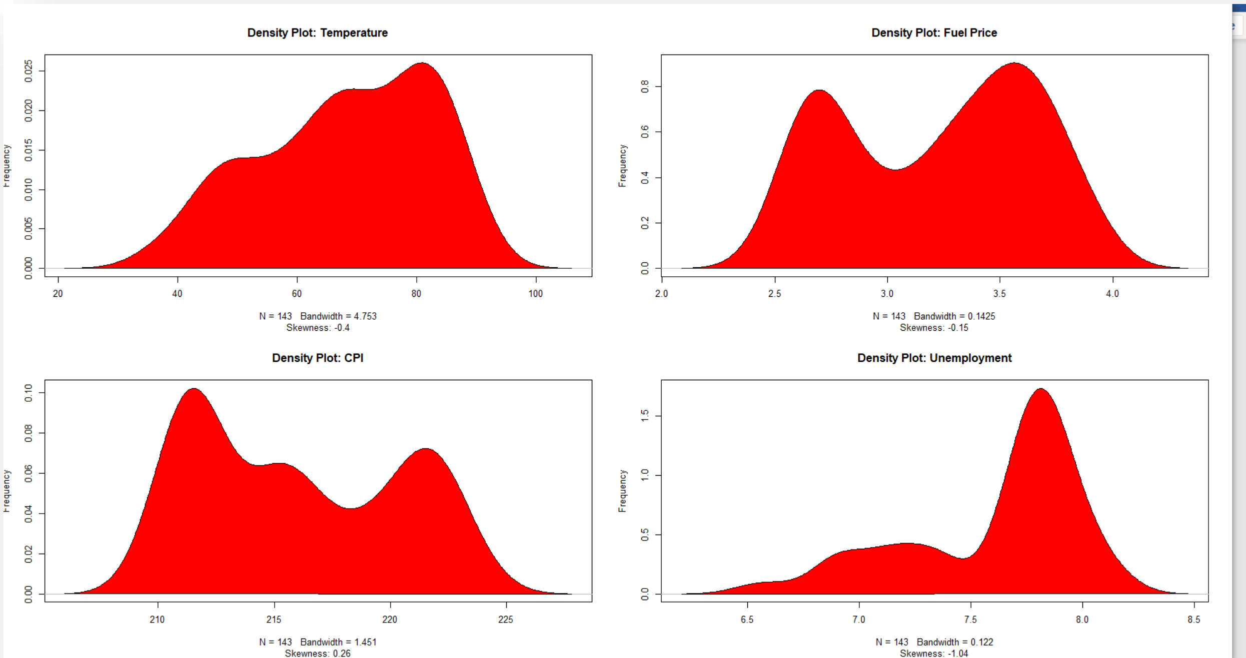
4. Next I checked for any outliers

```
#-------------------------------------BoxPlot - Check for outliers------------------------------------
par(mfrow=c(1, 4))  # divide graph area in 4 columns
boxplot(STORE1$Temperature, main="Temperature",col='red', sub=paste("Outlier rows: ", boxplot.stats(STORE1$Temperature)$out))
boxplot(STORE1$Fuel_Price, main="Fuel_Price",col='#E69F00', sub=paste("Outlier rows: ", boxplot.stats(STORE1$Fuel_Price)$out))
boxplot(STORE1$CPI, main="CPI",col='red', sub=paste("Outlier rows: ", boxplot.stats(STORE1$CPI)$out))
boxplot(STORE1$Unemployment, main="Unemployment",col='#E69F00', sub=paste("Outlier rows: ", boxplot.stats(STORE1$Unemployment)$out))
```



5. *Density plot – Check if the response variable is close to normality*

```r
#------------------------------------Density plot - Check if the response variable is close to normality------------

library(e1071)
par(mfrow=c(2, 2))  # divide graph area

plot(density(STORE1$Temperature), main="Density Plot: Temperature", ylab="Frequency",
     sub=paste("Skewness:", round(e1071::skewness(STORE1$Temperature), 2)))  # density plot for 'speed'
polygon(density(STORE1$Temperature), col="red")

plot(density(STORE1$Fuel_Price), main="Density Plot: Fuel Price", ylab="Frequency",
     sub=paste("Skewness:", round(e1071::skewness(STORE1$Fuel_Price), 2)))  # density plot for 'dist'
polygon(density(STORE1$Fuel_Price), col="red")

plot(density(STORE1$CPI), main="Density Plot: CPI", ylab="Frequency",
     sub=paste("Skewness:", round(e1071::skewness(STORE1$CPI), 2)))  # density plot for 'speed'
polygon(density(STORE1$CPI), col="red")

plot(density(STORE1$Unemployment), main="Density Plot: Unemployment", ylab="Frequency",
     sub=paste("Skewness:", round(e1071::skewness(STORE1$Unemployment), 2)))  # density plot for 'dist'
polygon(density(STORE1$Unemployment), col="red")
```



<u>Next, we standardize our dataset.</u> The purpose of data standardization is to make our data consistent and clear. Consistent is ensuring that the output is reliable so that related data can be identified using common terminology and format. Clear is to ensure that the data can be easily understood by those who are not involved with the data maintenance process.

```
#---------------------------DATA STANDARDIZATION------------------------------

Store_1 <- STORE1 %>% mutate_all(~(scale(.) %>% as.vector))
View(Store_1)

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

	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
1	0.566906444	-0.2732438	-1.82427339	-1.51575083	-1.126328975	1.2914187
2	0.555793151	3.6341427	-2.09093096	-1.57191581	-1.092815924	1.2914187
3	0.363530538	-0.2732438	-1.99128524	-1.65148287	-1.082019719	1.2914187
4	-0.933043282	-0.2732438	-1.52112585	-1.54149311	-1.075009659	1.2914187
5	-0.002934449	-0.2732438	-1.53024834	-1.39171983	-1.067999600	1.2914187
6	-0.741904334	-0.2732438	-0.73799468	-1.29343111	-1.060989540	1.2914187
7	-0.530505195	-0.2732438	-0.96325015	-1.16940010	-1.098914630	1.2914187
8	-0.967006893	-0.2732438	-1.18289178	-1.14131761	-1.144328924	1.2914187
9	0.254543448	-0.2732438	-0.42361945	-1.17174031	-1.189743195	0.5148691
10	-0.063122318	-0.2732438	-0.17169822	-1.05238972	-1.235157465	0.5148691
11	-0.571904593	-0.2732438	-0.13941862	-0.96346183	-1.265991938	0.5148691
12	-1.051464741	-0.2732438	-0.24327473	-0.99388453	-1.277386664	0.5148691
13	-0.834485491	-0.2732438	-0.06293001	-1.02898765	-1.288781389	0.5148691
14	0.312158500	-0.2732438	0.29775944	-0.90027623	-1.300176115	0.5148691
15	-0.391156541	-0.2732438	0.45424532	-0.85581228	-1.300760433	0.5148691

BUILD MODEL: -

LINEAR REGRESSION MODEL WITH CROSS VALIDATION

I used Linear Regression model with Cross validation feature. CV provides the ability to estimate model performance on unseen data not used while training.

- First step is to split the dataset into training and testing dataset. I did the ratio of 70:30 for split.

```r
#------------------------------------BUILDING MODELS   -------------------------

# setting seed to generate a
# reproducible random sampling
set.seed(123)


# Data split


sample = sample.split(STORE1, SplitRatio = 0.7)
train = subset(STORE1, sample == T)
test = subset(STORE1, sample == F)
dim(train)
dim(test)
View(train)
View(test)
```

```r
#-------------------------LINEAR REGRESSION USING CROSS VALIDATION---------------------------------

train_control <- trainControl(method = "cv",
                              number = 10)

lm_model <- train(Weekly_Sales~., data = train,
                  trControl = train_control,
                  method = "lm")

summary(lm_model)
```

```
>
> summary(lm_model)

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min      1Q  Median      3Q     Max 
-308398  -76186  -17049   64952  571303 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)   -4772632    1950339  -2.447  0.01637 *  
Holiday_Flag     87039      58050   1.499  0.13732    
Temperature      -1383       1023  -1.351  0.17997    
Fuel_Price      -69919      51553  -1.356  0.17845    
CPI              26035       7531   3.457  0.00084 ***
Unemployment    132777      65444   2.029  0.04546 *  
---
Signif. codes:  
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 132400 on 89 degrees of freedom
Multiple R-squared:  0.1967,	Adjusted R-squared:  0.1516 
F-statistic: 4.359 on 5 and 89 DF,  p-value: 0.001353
```

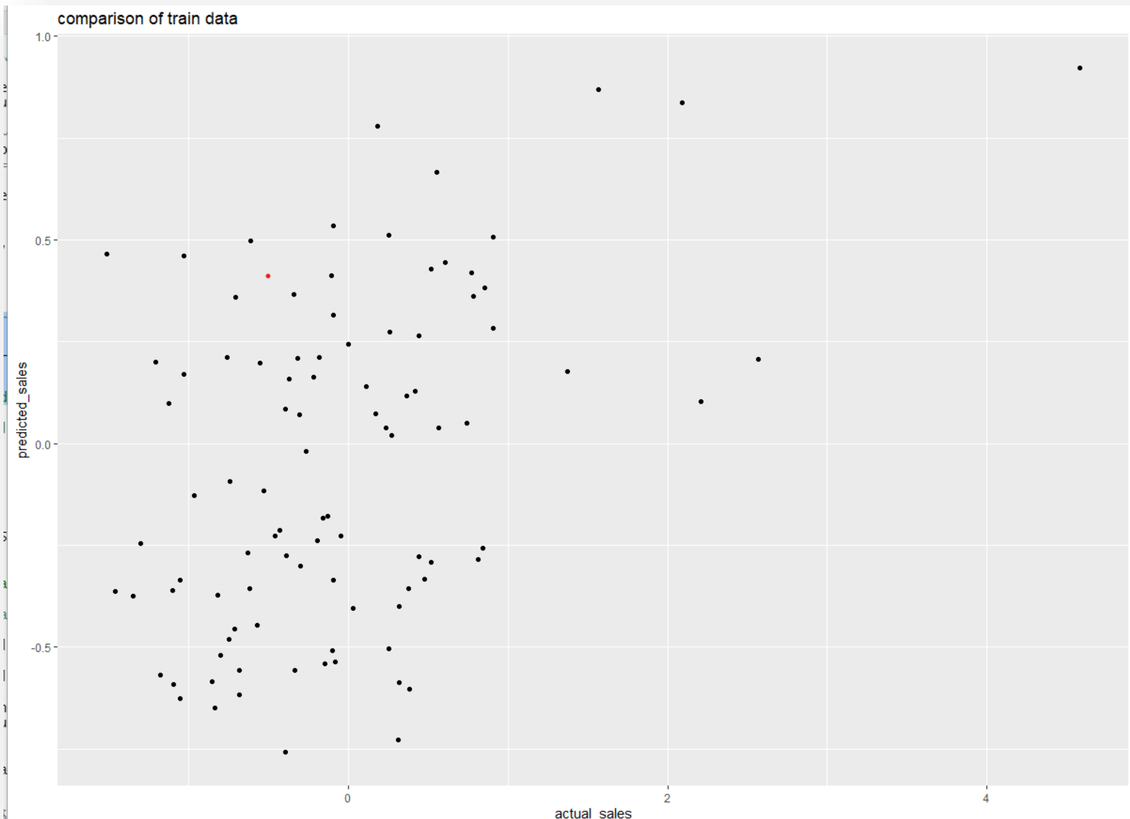
```
> y_pred_train = predict(lm_model, newdata = train)
>
> length(y_pred_train)
[1] 95
> y_pred_train
          1           2           3           6           7           8           9
 0.03829408  0.66509419  0.11757501 -0.09410328 -0.11693914 -0.12754469 -0.50430947
         12          13          14          15          18          19          20
-0.62481392 -0.64915379 -0.72766019 -0.75637518 -0.60194995 -0.53637896 -0.55550201
         21          24          25          26          27          30          31
-0.58392844 -0.61757466 -0.59021995 -0.56859303 -0.58744608 -0.55620172 -0.50783611
         32          33          36          37          38          39          42
 0.07187863 -0.51952472 -0.30032978 -0.35696498 -0.37503356 -0.36111176 -0.22731697
         43          44          45          48          49          50          51
 0.20700564 -0.22725078 -0.28527062  0.19910654 -0.45402651 -0.33425264 -0.36392918
         54          55          56          57          60          61          62
 0.44440889 -0.25791446 -0.26797893 -0.29047613 -0.33590779 -0.27448125 -0.35569237
         63          66          67          68          69          72          73
-0.40404847 -0.33391007 -0.40052766 -0.37149969 -0.44514260 -0.53982079 -0.47992236
         74          75          78          79          80          81          84
-0.21213717 -0.17936740 -0.24435262 -0.27777888 -0.23905732 -0.18352998  0.53367810
         85          86          87          90          91          92          93
-0.02025859  0.09877633  0.17036537  0.36566880  0.35861520  0.50613934  0.51203024
```

Comparison of Training and Test Data

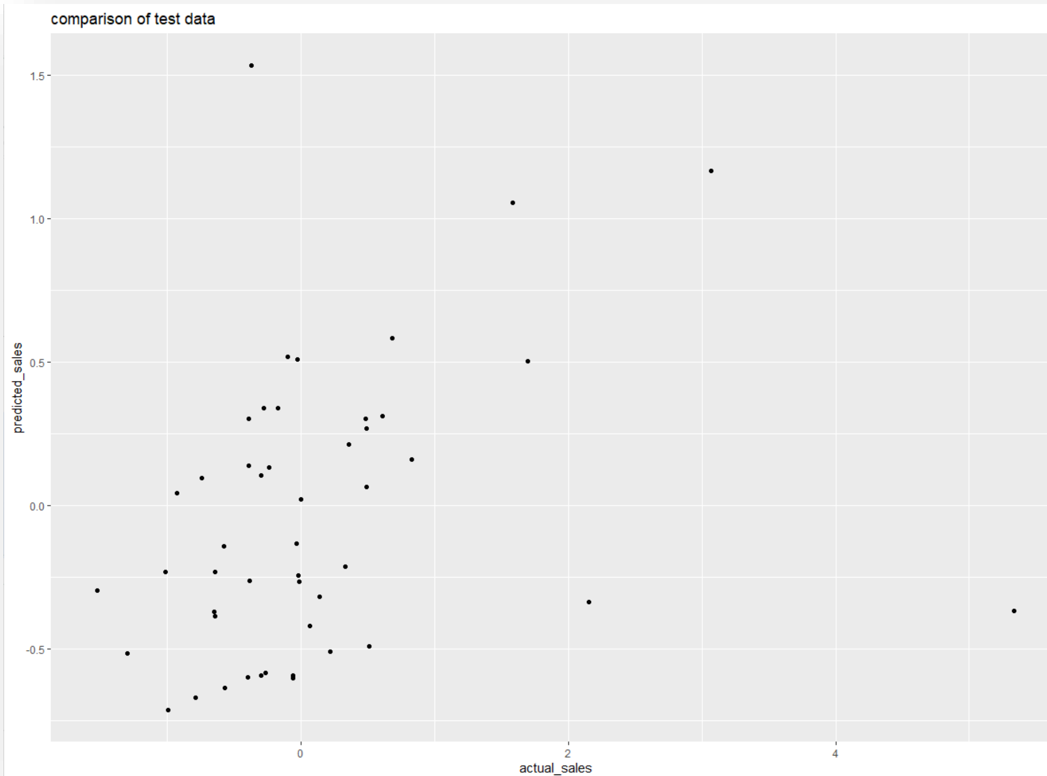
```
# Visualizing the training set results
ggplot() +
  geom_point(aes(x=train$Weekly_Sales,y=y_pred_train) +
  xlab('actual_sales') +
  ylab('predicted_sales')+
  ggtitle('comparison of train data'))

# Visualizing the test set results

y_pred_test = predict(lm_model, newdata = test)

y_pred_test

ggplot() +
  geom_point(aes(x=test$Weekly_Sales,y=y_pred_test)) +
  xlab('actual_sales') +
  ylab('predicted_sales')+
  ggtitle('comparison of test data')
```

Train Data

TR



Test data



Validate the accuracy of model.

```
### Parameters to validate the accuracy of the model and improvise.

MAPE(y_pred_test,test$Weekly_Sales)

RMSE(y_pred_test,test$Weekly_Sales)

actuals_preds <- data.frame(cbind(actuals=test$Weekly_Sales, predicteds=y_pred_test))
# make actuals_predicteds dataframe.
correlation_accuracy <- cor(actuals_preds)
correlation_accuracy

head(actuals_preds)

min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, max))
min_max_accuracy <- min_max_accuracy * 100

mape <- mean(abs((actuals_preds$predicteds - actuals_preds$actuals))/actuals_preds$actuals)
mape
```

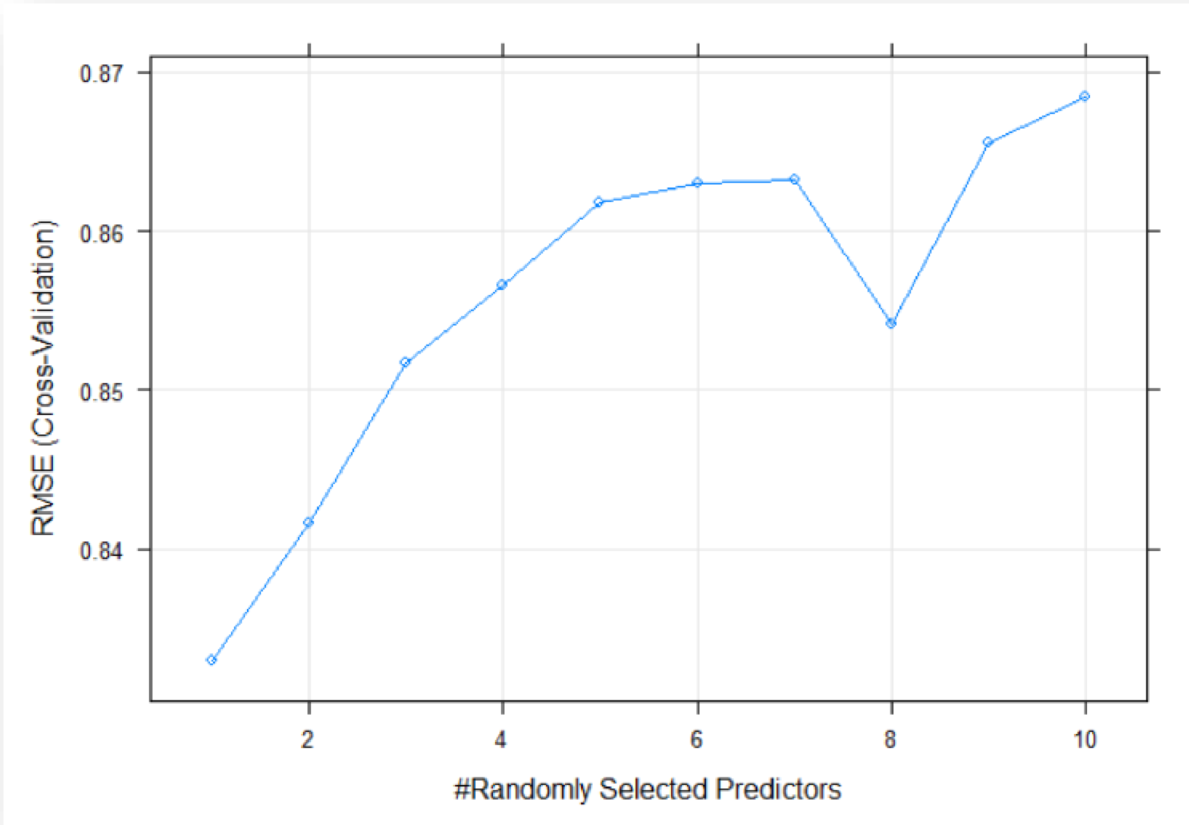
```
> MAPE(y_pred_test,test$Weekly_Sales)
[1] 2.755393
>
> RMSE(y_pred_test,test$Weekly_Sales)
[1] 1.109936
>
> actuals_preds <- data.frame(cbind(actuals=test$Weekly_Sales, predicteds=y_pred_test))
> # make actuals_predicteds dataframe.
> correlation_accuracy <- cor(actuals_preds)
> correlation_accuracy
            actuals predicteds
actuals    1.000000   0.279903
predicteds 0.279903   1.000000
>
> head(actuals_preds)
        actuals  predicteds
4  -0.933043282  0.04218778
5  -0.002934449  0.01974300
10 -0.063122318 -0.59198438
11 -0.571904593 -0.63548958
16 -0.997573802 -0.71186347
17 -0.789805367 -0.67061900
>
> min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, max))
> min_max_accuracy <- min_max_accuracy * 100
> min_max_accuracy
[1] 79.63442
>
> mape <- mean(abs((actuals_preds$predicteds - actuals_preds$actuals))/actuals_preds$actuals)
> mape
[1] -1.691429
>
```

We achieved accuracy of 79.63% with linear regression model.

Random Forest Model

The second model I built was using Random Forest algorithm. I used the cross validation model with random forest to achieve optimal result.

```
#----------------------------------RANDOM FOREST MODEL  --------

set.seed(1234)
tuneGrid <- expand.grid(.mtry = c(1: 10))

rm_model <- train(Weekly_Sales~.,
                 data = train,
                 method = "rf",
                 tuneGrid = tuneGrid,
                 trControl = trControl,
                 importance = TRUE,
                 nodesize = 14,
                 ntree = 300)
print(rm_model)
plot(rm_model)

test.features = subset(test, select=-c(Weekly_Sales))
test.target = subset(test, select=Weekly_Sales)[,1]

predictions = predict(rm_model, newdata = test.feature)
```


RMSE (Cross-Validation)
0.87
0.86
0.85
0.84
2
4
6
8
10
#Randomly Selected Predictors

```
Random Forest

95 samples
 5 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 86, 86, 85, 84, 86, 86, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
   1    0.8329754  0.1129969  0.6256340
   2    0.8416648  0.1276761  0.6295883
   3    0.8517369  0.1167243  0.6375178
   4    0.8565556  0.1333856  0.6362010
   5    0.8617449  0.1301845  0.6419391
   6    0.8630288  0.1261085  0.6436063
   7    0.8632295  0.1221776  0.6423347
   8    0.8541077  0.1318110  0.6361722
   9    0.8655030  0.1174441  0.6452284
  10    0.8684611  0.1151875  0.6477853

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 1.
>
```

```
> MAPE(predictions,test.target)
[1] 2.021534
>
> RMSE(predictions,test.target)
[1] 1.120096
>
> actuals_preds <- data.frame(cbind(actuals=test.target, predicteds=predictions))
> # make actuals_predicteds dataframe.
> correlation_accuracy <- cor(actuals_preds)
> correlation_accuracy
             actuals predicteds
actuals    1.0000000  0.2182725
predicteds 0.2182725  1.0000000
>
> head(actuals_preds)
        actuals   predicteds
4  -0.933043282  0.097811874
5  -0.002934449  0.002421903
10 -0.063122318 -0.446519499
11 -0.571904593 -0.441110361
16 -0.997573802 -0.368528276
17 -0.789805367 -0.377667928
>
> min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, m
ax))
>
> min_max_accuracy
[1] 1.144863
>
> mape <- mean(abs((actuals_preds$predicteds - actuals_preds$actuals))/actuals_pred
s$actuals)
> mape
[1] -1.235475
> |
```

RMSE was used to select the optimal model using the smallest value.

The final value used for the model was mtry = 1.

CONCLUSION –

Best accuracy is achieved by Random Forest algorithm -83.29%