

# **Advances in Data Sciences**

## **Final Project Report**

### **Team 8**

Dhaval Sanghavi

Sheetal Singh

Tanisha Jain

## INDEX

1. Rossmann Store Sales Analysis
  - 1.1 Problem Statement
  - 1.2 Background and Summary
  - 1.3 Dataset
  - 1.4 Evaluation of Criteria
  - 1.5 Impact of Solution
  - 1.6 Power BI Analysis
    - 1.6.1 Visualization on Raw Data
    - 1.6.2 Visualization on Cleaned Data
  - 1.7 Data Cleansing
  - 1.8 Regression Models
    - 1.8.1 Overall Design
    - 1.8.2 Azure Models
    - 1.8.3 Boosted Decision Tree Regression Model
    - 1.8.4 Bayesian Linear Regression
    - 1.8.5 Decision Forest Regression
    - 1.8.6 Linear Regression
    - 1.8.7 Neural Network Regression
    - 1.8.8 Poisson Regression
    - 1.8.9 ARIMA Model
    - 1.8.10 Summary
    - 1.8.11 Conclusion
    - 1.8.12 Web Service
  - 1.9 Classification Model
    - 1.9.1 Overall Design
    - 1.9.2 Azure Models
    - 1.9.3 Two Class Support Vector Machine
    - 1.9.4 Two Class Averaged Perceptron
    - 1.9.5 Two Class Bayes Point Machine
    - 1.9.6 Two Class Boosted Decision Tree

1.9.7 Two Class Decision Forest

1.9.8 Two Class Decision Jungle

1.9.9 Two Class Logistic Regression

1.9.10 Two Class Neural Network

1.9.11 Summary

1.9.12 Conclusion

1.9.13 Web Service

1.10 Web Application

1.10.1 The Algorithm: K-Means clustering

1.10.2 Web Service

1.10.3 Cluster Formation

1.11 Web Application

1.12 Integration of Power Bi with web application

1.13 References

# **Rossmann Store Sales Analysis**

## **1.1 Problem Statement**

Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied. In this case study, we focus on daily sales for 1,115 stores located across Germany.

## **1.2 Background and Summary**

Rossmann is Germany's second-largest drug store chain, with other 3,000 stores in 7 European countries. Rossmann Store sells prescription drugs and a wide assortment of general merchandise, including over-the-counter drugs, beauty products and cosmetics. It also provides healthcare services through its more than 1000 MinuteClinic medical clinics as well as their Diabetes Care Centers. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

## **1.3 Dataset**

Most of the fields are self-explanatory. The following are descriptions for those that aren't.

**Id** - an Id that represents a (Store, Date) duple within the test set

**Store** - a unique Id for each store

**Sales** - the turnover for any given day (this is what you are predicting)

**Customers** - the number of customers on a given day

**Open** - an indicator for whether the store was open: 0 = closed, 1 = open

**StateHoliday** - indicates a state holiday. Normally all stores, with few exceptions, are closed on state holidays. Note that all schools are closed on public holidays and weekends. a = public holiday, b = Easter holiday, c = Christmas, 0 = None

**SchoolHoliday** - indicates if the (Store, Date) was affected by the closure of public schools

**StoreType** - differentiates between 4 different store models: a, b, c, d

**Assortment** - describes an assortment level: a = basic, b = extra, c = extended

**CompetitionDistance** - distance in meters to the nearest competitor store

**CompetitionOpenSince[Month/Year]** - gives the approximate year and month of the time the nearest competitor was opened

**Promo** - indicates whether a store is running a promo on that day

**Promo2** - Promo2 is a continuing and consecutive promotion for some stores: 0 = store is not participating, 1 = store is participating

**Promo2Since[Year/Week]** - describes the year and calendar week when the store started participating in Promo2

**PromoInterval** - describes the consecutive intervals Promo2 is started, naming the months the promotion is started anew. E.g. "Feb,May,Aug,Nov" means each round starts in February, May, August, November of any given year for that store

## 1.4 Evaluation Criteria

Predictions are evaluated on the Root Mean Square Percentage Error(RMSPE). Lower the score better will be the prediction.

$$\text{RMSPE} = \sqrt{\frac{1}{n} \sum_{i=1}^n \left( \frac{y_i - \hat{y}_i}{y_i} \right)^2},$$

where Sales denotes the sales of a single store on a single day and PredSales denotes the corresponding prediction. Store with 0 sales is ignored in scoring.

## 1.5 Impact of Solution

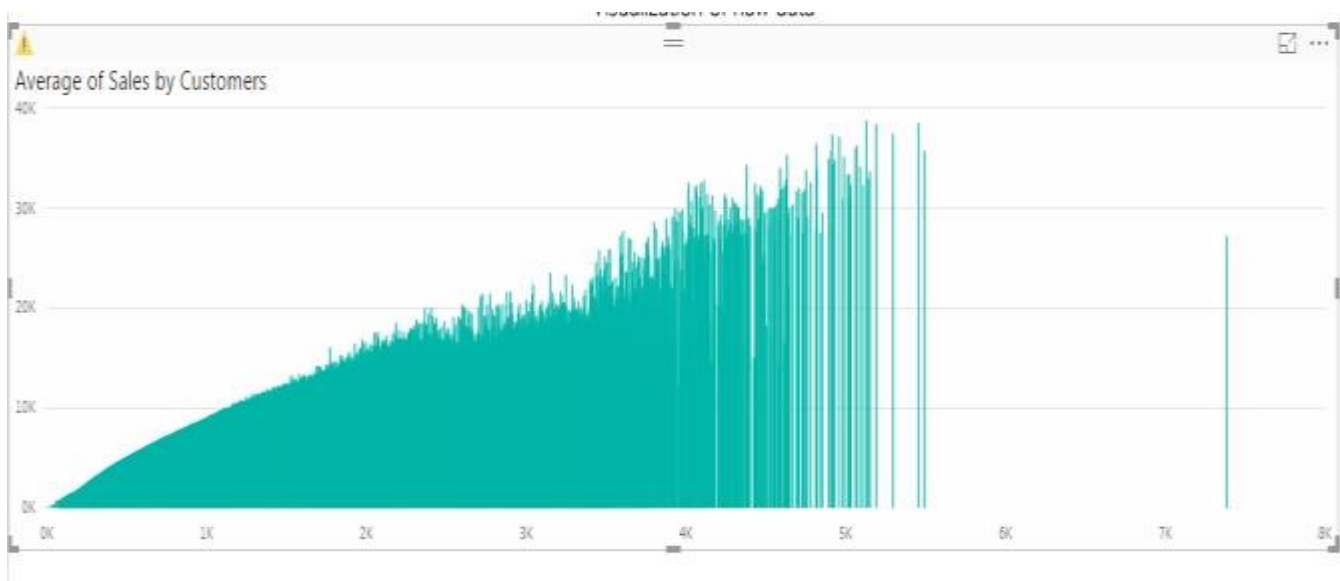
- Better management of staff schedules.
- Provide enough time to store managers to focus on customers and their teams.
- Increase efficiency of employees.

## 1.6 Power BI Analysis

### 1.6.1 Visualization of Raw Input File

#### Graph1: Average Sales as per customers

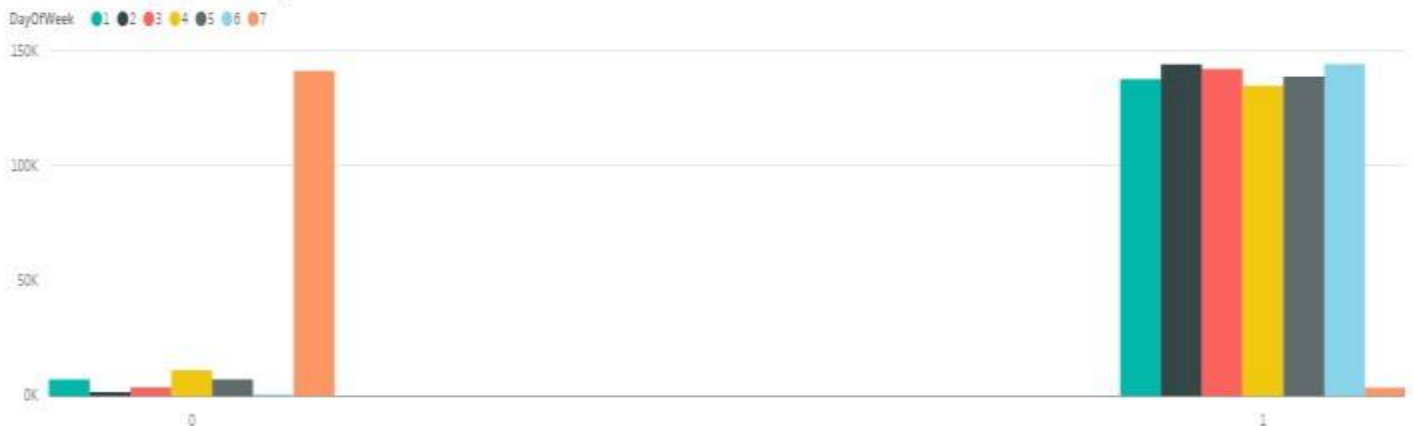
This graph shows that Average sales vary as per the number of customers in the store. Sales depends on various other factors like date, Day of Week, it's a holiday or not, etc.



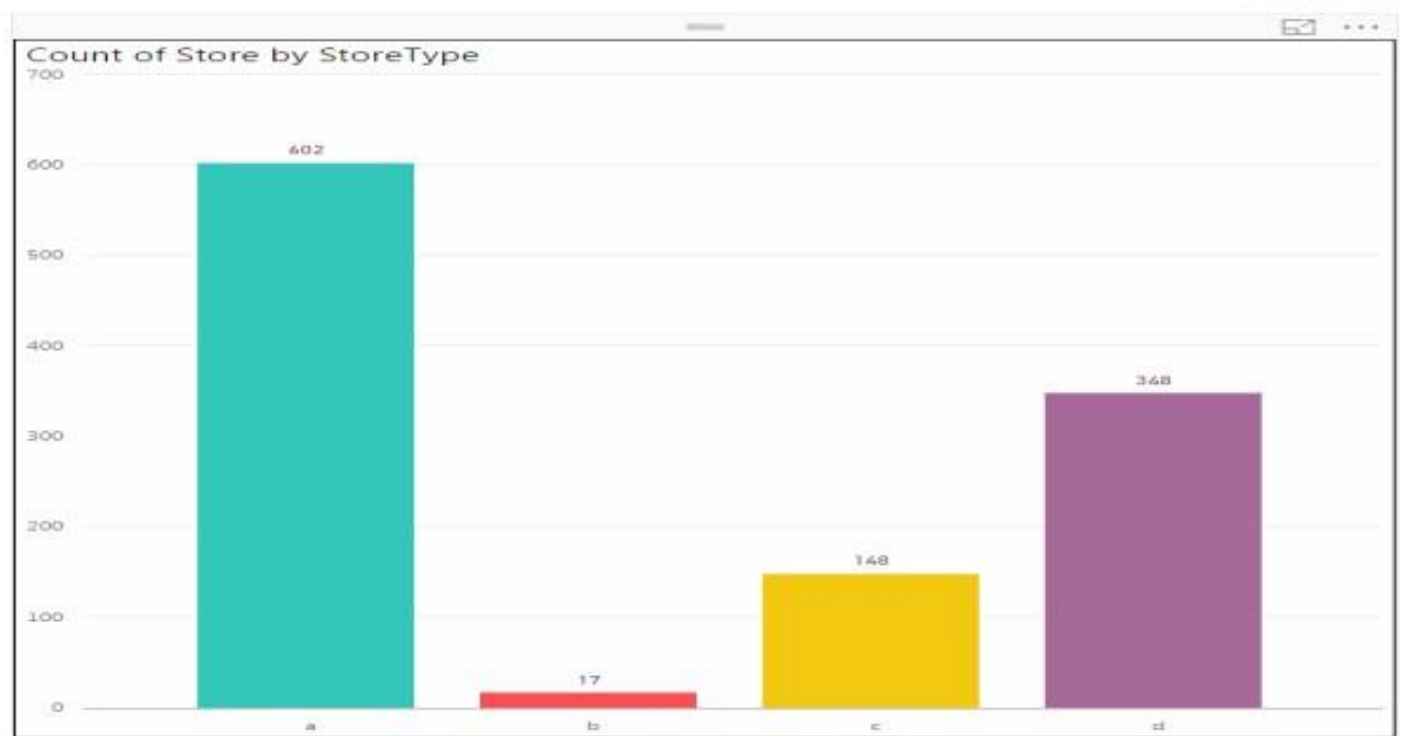
**Graph 2: Count of Sales by Open and DayofWeek**

This graph represents how sales varies as per 'Open' status and which Day of Week it is. Sales is more on weekends as well as when the 'Open' status is 1.

Count of Sales by Open and DayOfWeek

**Graph 3: Count of Store by Store Type**

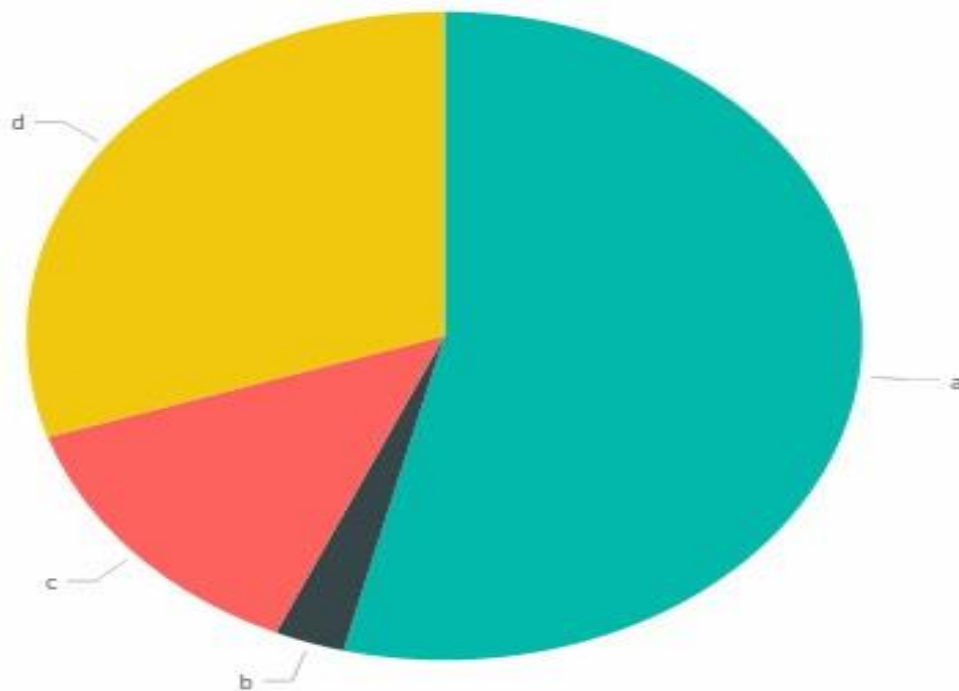
This graph represents the number of stores as per the Store Type. Various Store Types are a,b,c and d.



**Graph 4: Sales by Store Type**

This graph represents Average of Sales as per each store. So on an average Store Type b have maximum sales.

Sales by StoreType

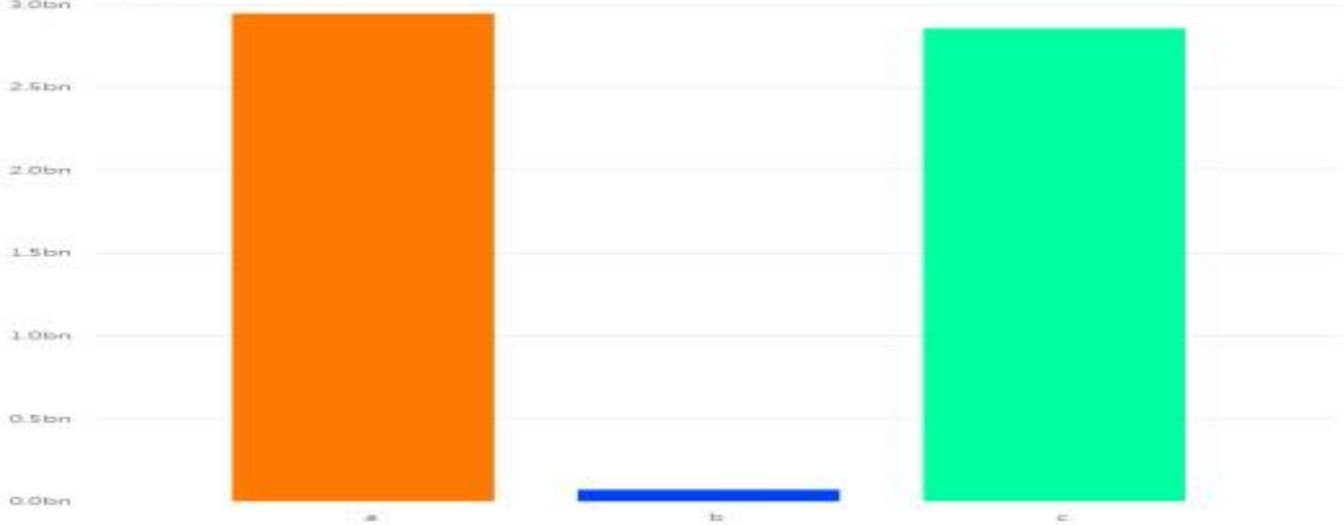




### Graph 5: Count of Sales by Assortment

Assortment basically specifies the type of store i.e. if its Basic,extra or extender. This graph shows Sales as per Assortment Type.

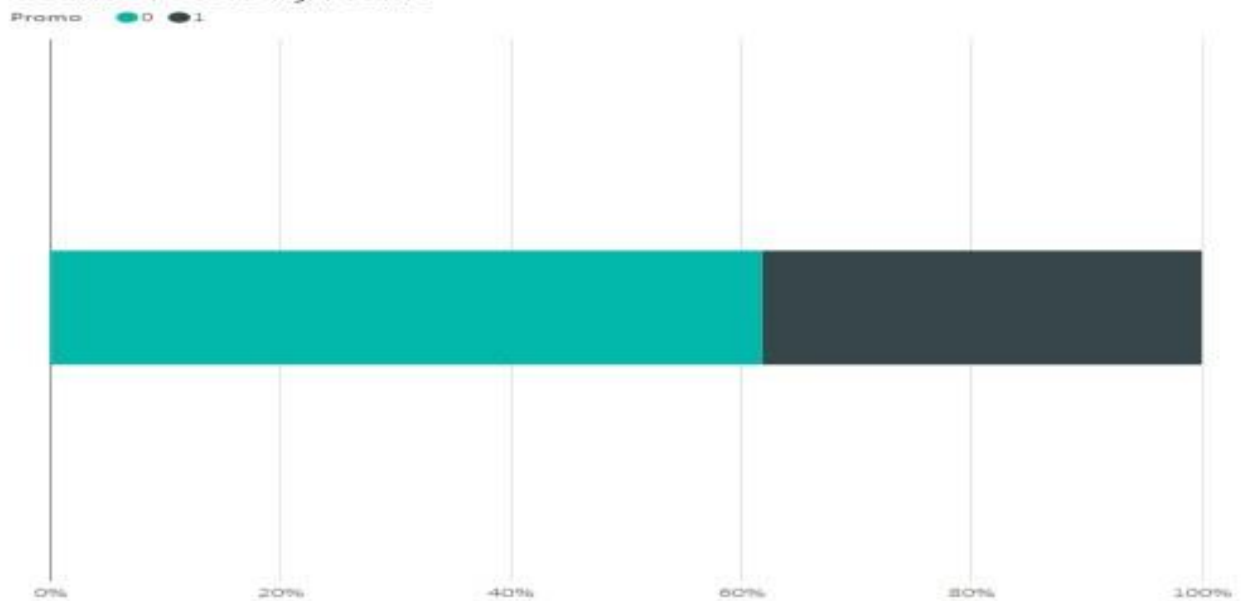
Sales by Assortment



### Graph 6: Count of Store by Promo

This graph displays the count of store by promo. It shows that there are more stores which do not offer any promotions.

Count of Store by Promo

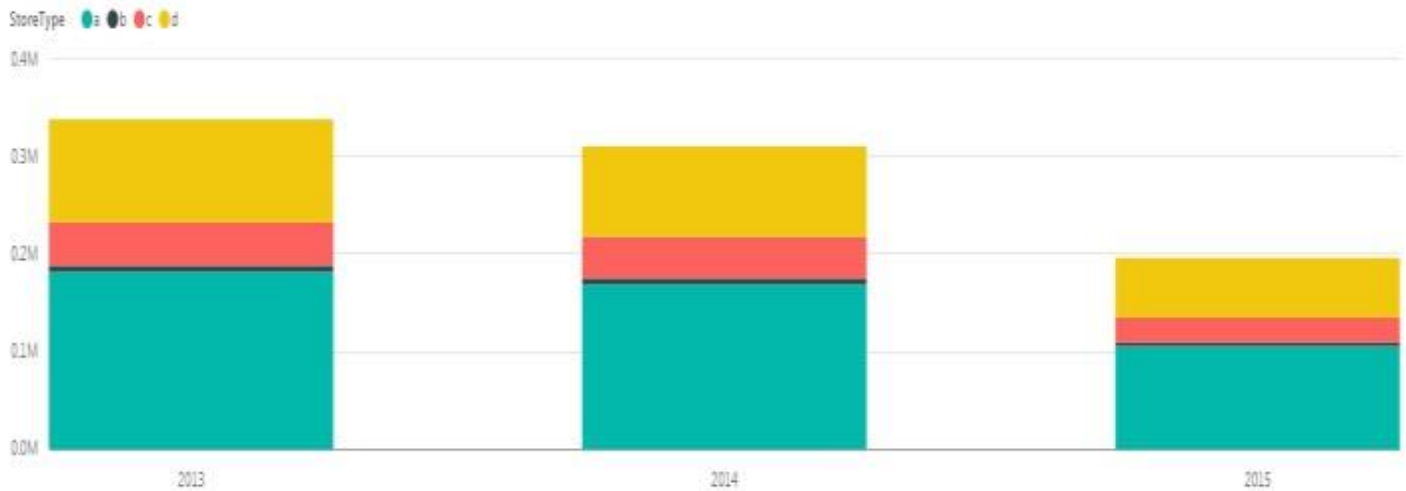


## 1.6.2 Visualization on Cleaned File

### Graph 1: Count of Sales by Year and Store Type

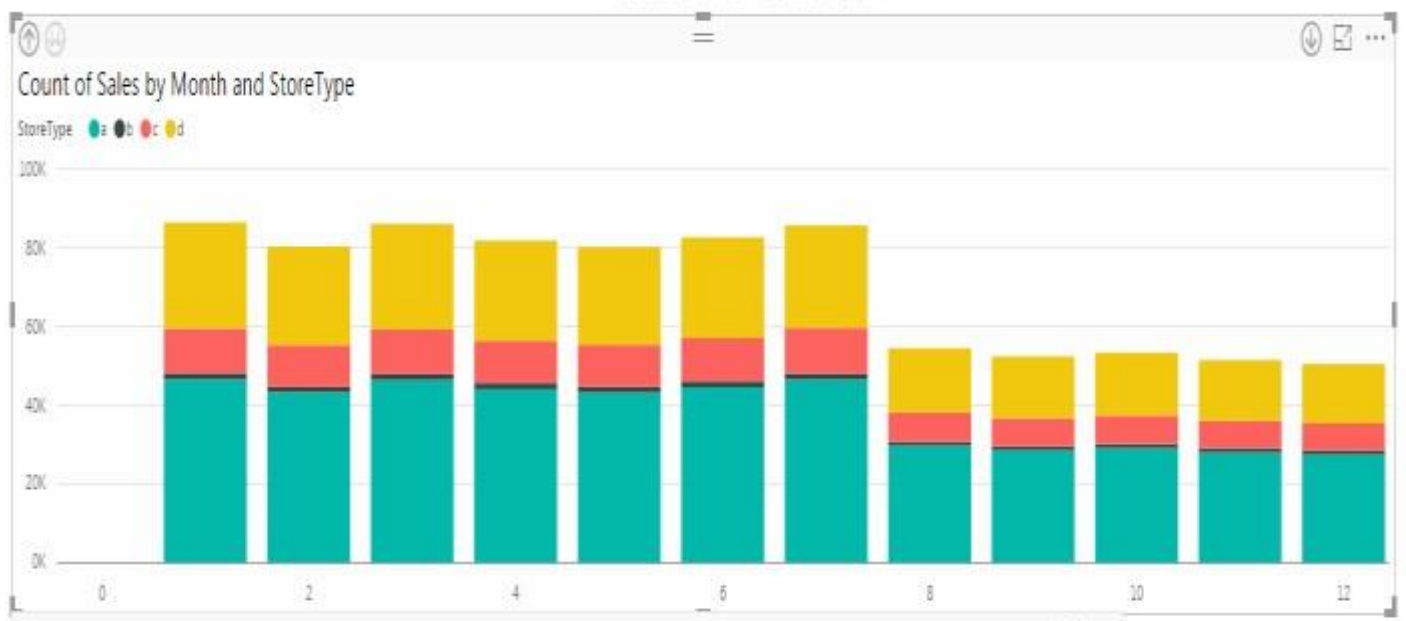
This graph shows Sales count for every year as per the Store Type.

Count of Sales by Year and StoreType



### Graph 2: Count of Sales by Month and Store Type

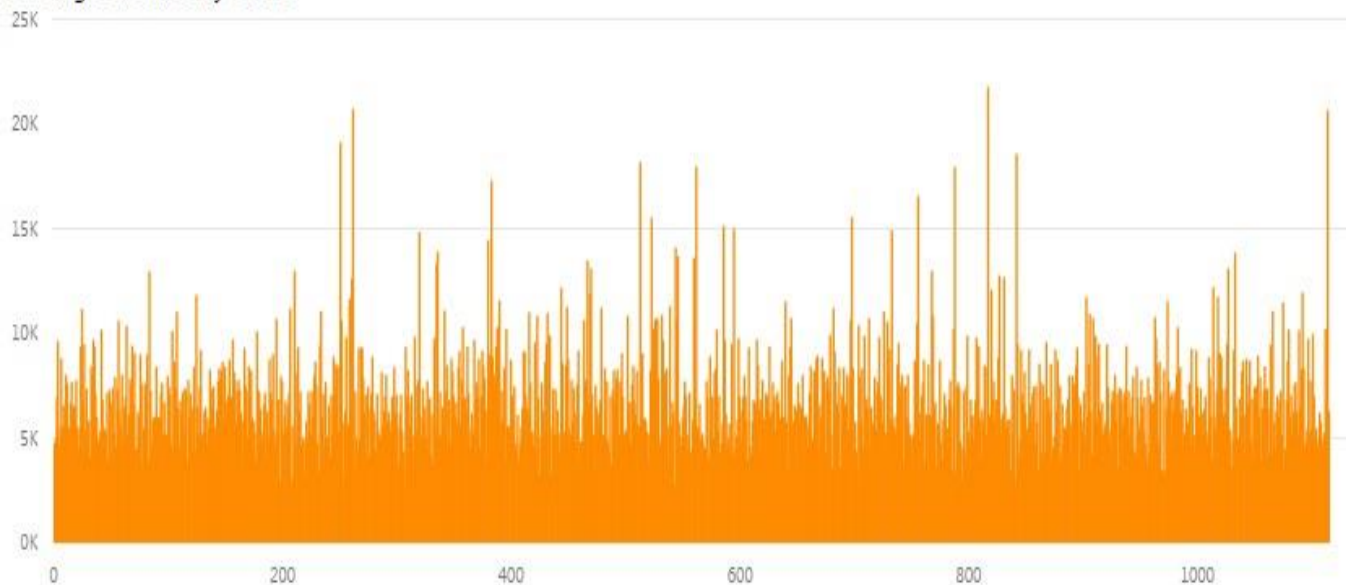
This graph shows Sales count for every month as per the Store Type.



### Graph 3: Sales by Store

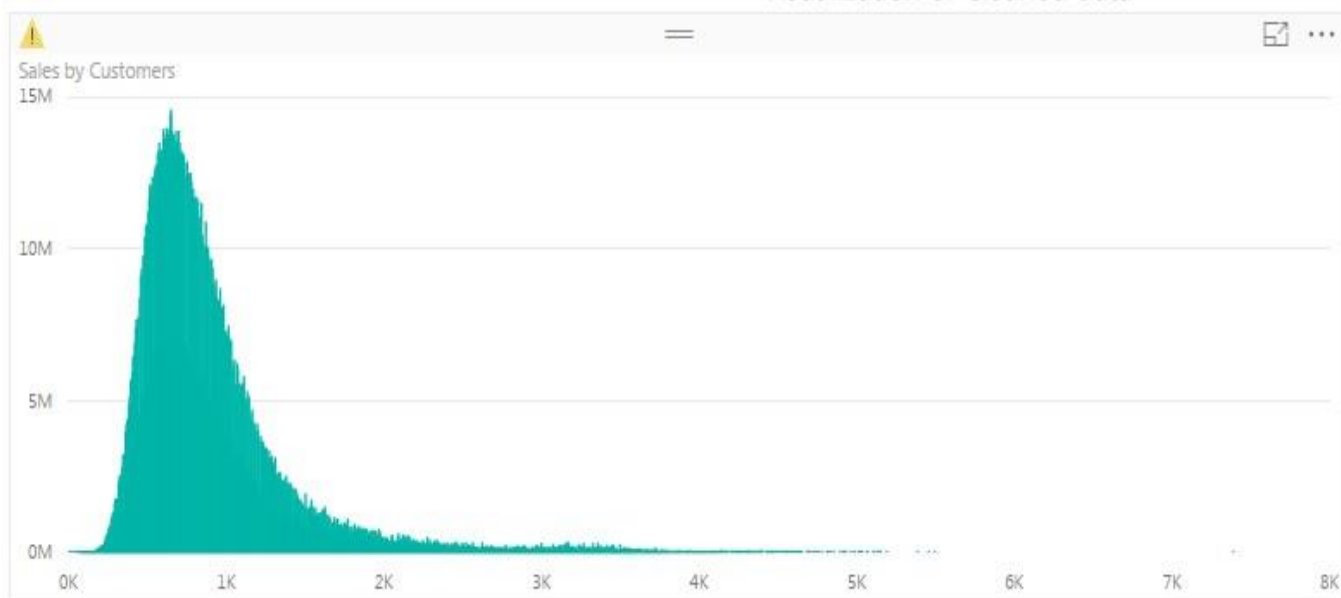
This graph shows Average Sales by each Store.

Average of Sales by Store



### Graph 4: Sales by Customers

This graph shows the Sales count as per number of Customers.



## Graph 5: Sales by State Holiday

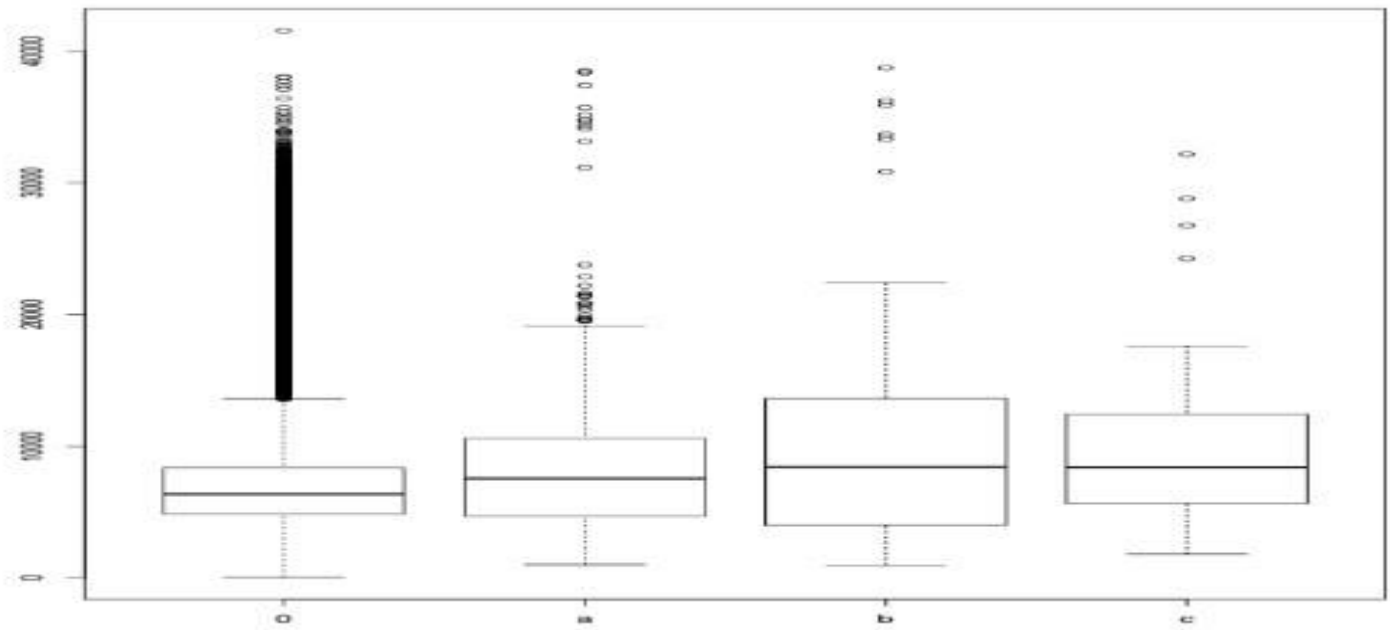
**This graph shows the Count of Sales as per the type of State Holidays.**

Count of Sales by StateHoliday

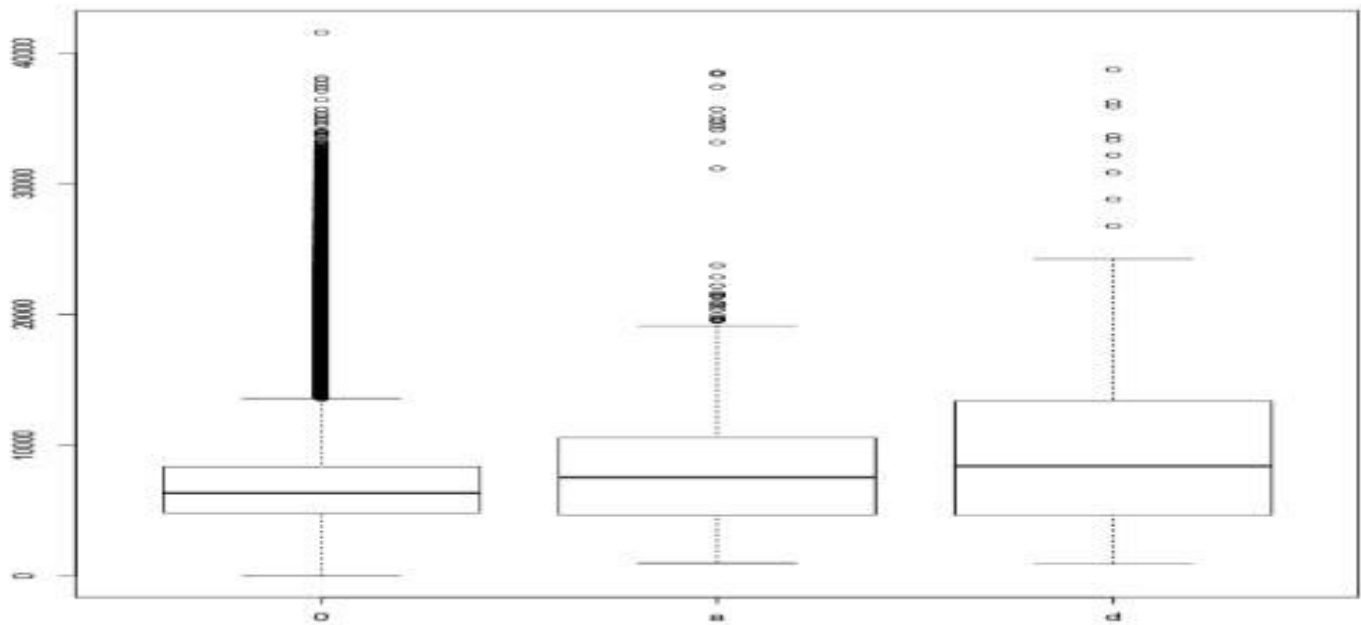


**a= Public Holiday, 0= no holiday, d= Combination of Easter Holiday and Christmas Holiday**

**Note:** We have combined StateHoliday for b=Easter Holiday and c=Christmas Holiday as d because State Holiday a, b, c's sales distribution is not similar. However, State Holiday==b only has 145 data points. StateHoliday==c only has 71 data points. Since the training data points are not large, we combined State Holiday b and c as one category as depicted in the boxplot below.



**Boxplot of Sales as per StateHoliday in uncleaned dataset**

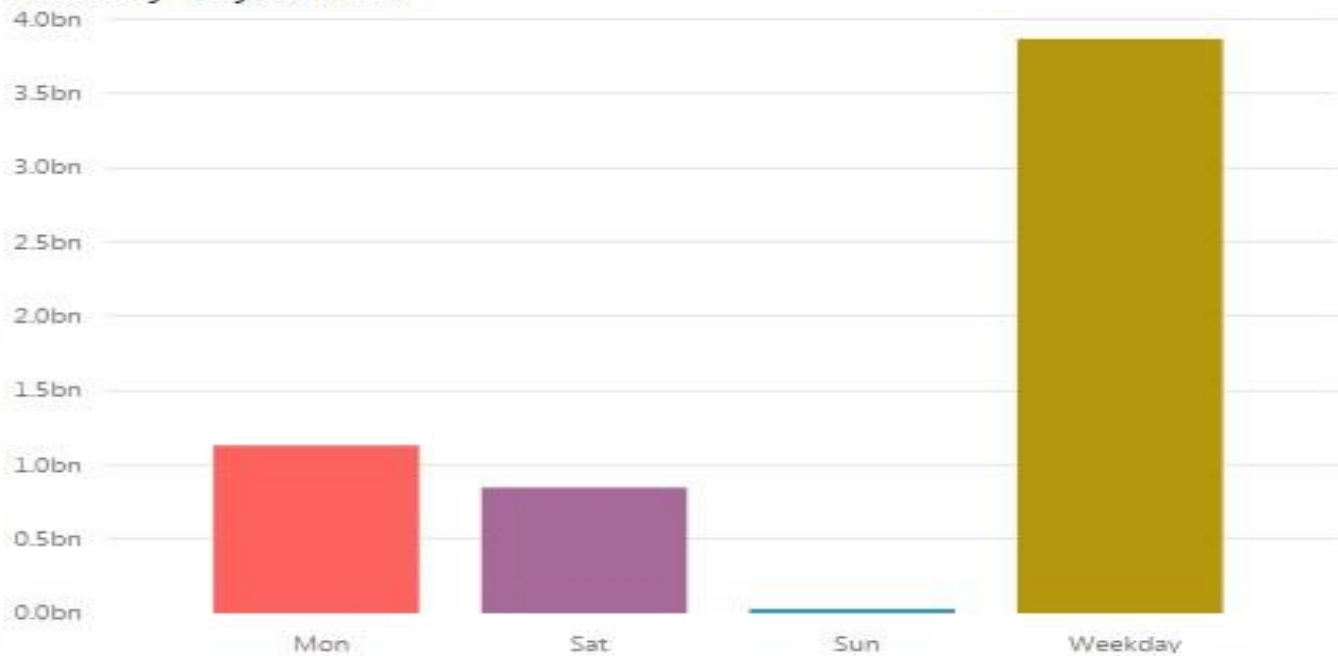


**Boxplot of Sales as per StateHoliday in cleaned dataset**

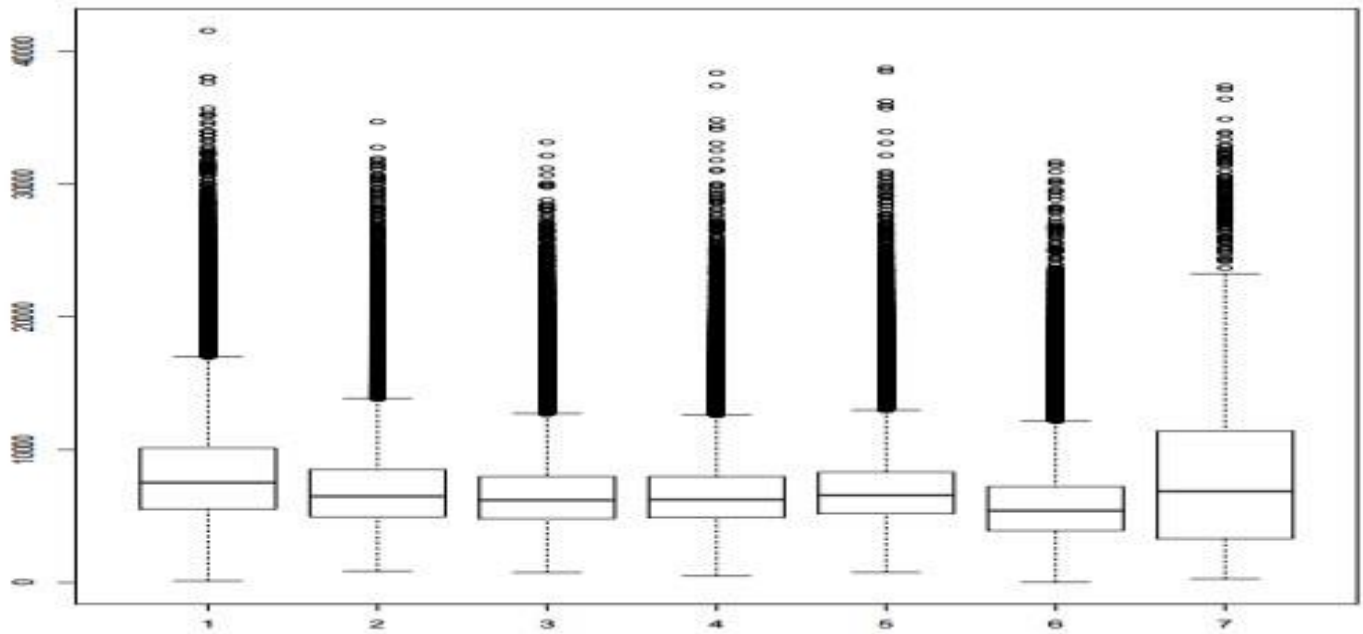
## Graph 6: Sales by DayofWeek

This graph depicts Sales by each day of week.

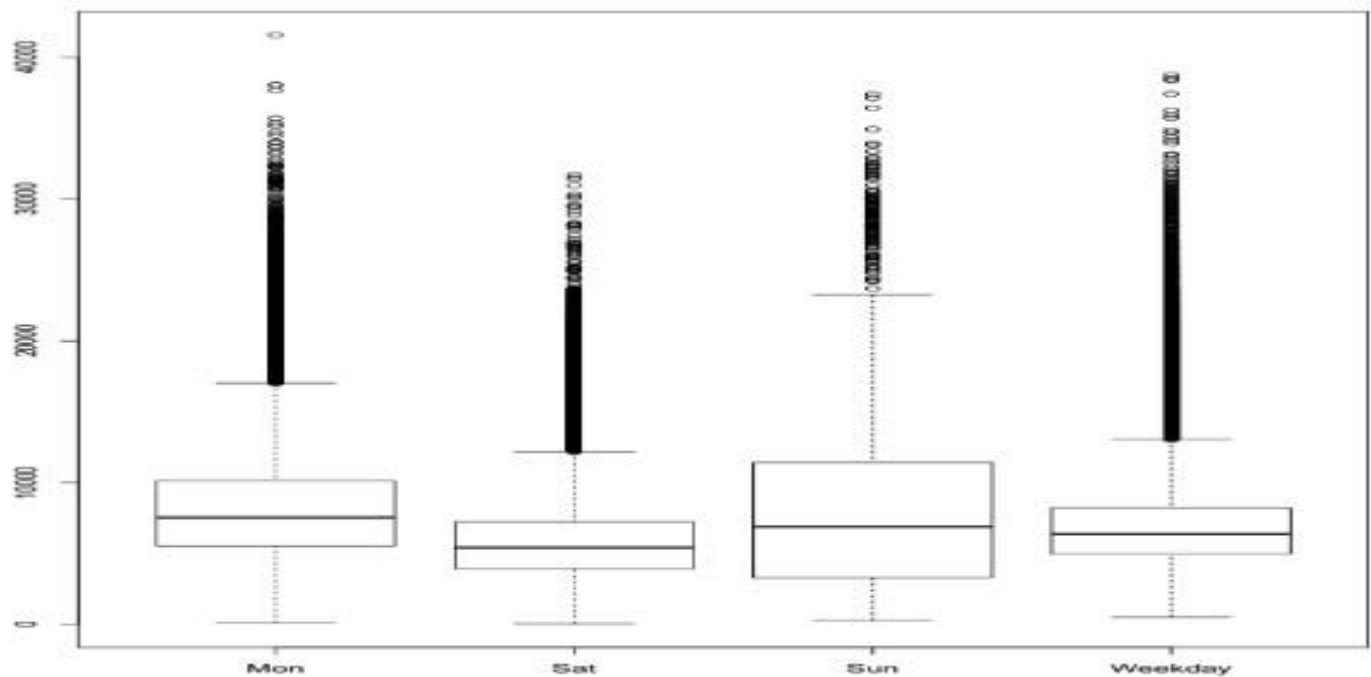
Sales by DayOfWeek



**Note:** We have combined Tue, Wed, Thurs and Fri as Weekday because Tue through Fri Sales distributions are very close. Mon, Sat and Sun's Sales distributions are unique. In dataset, DayofWeek is represented as numeric number 1-7. From intuitive, we know that there is no linear relationship from 1-7 number to Sales data. We treat DayofWeek as four factors, Mon, Weekday(Tue-Fri), Sat, Sun as depicted in the boxplot below.



Boxplot of Sales as per DayofWeek in uncleaned dataset



Boxplot of Sales as per DayofWeek in cleaned dataset

## Graph 7: Average of Sales by Promo2Indicator

This graph indicates average of sales as per Promotion-2 indicator.



**Note:** We combined Promo2, Promo2SinceWeek, Promo2SinceYear and Promointerval to a Promo2indicator in historical sales data. The indicator indicates on a certain day whether a certain store is on promotion 2.

## 1.7 Data Cleansing

Data cleansing is divided into two parts:

### RStudio:

- Read the train.csv file.
- Read the store.csv file.
- Merge data from train and store file.
- Write merged data to merged\_data\_train\_updated.csv file.



### #ROSSMANN SALES DATASET : Data Cleansing

#Train dataset as input

```
rossmann_data<-read.csv(file.choose(), header=TRUE)
```

#Store dataset as input

```
store_data<-read.csv(file.choose(), header=TRUE)
```

#Merging Train dataset with the Store dataset

```
merged_data<-merge(rossmann_data,store_data,by='store',all.x=T)
```

#write merged data to csv file

```
write.csv(merged_data,"merged_data_train_updated.csv",row.names=FALSE)
```

### Microsoft Azure ML:

Read merged\_data\_train\_updated.csv file.

Remove outliers.

*Remove the rows where store is open and sales value is zero.*

*Remove the rows where sales<0 when customers visit the store.*

#Removing rows where sales=0 and Open=1

```
order_data<-merged_data[!(merged_data$Open==1 & merged_data$Sales==0),]
```

#Remove the rows where sales<0 when customers visit the store

```
order_data<-merged_data[(merged_data$Sales>0),]
```

Extract Day, month, year and week from each date.

Rename the DayofWeek column.

**Note:** We have combined Tue, Wed, Thurs and Fri as Weekday because Tue through Fri Sales distributions are very close. Mon, Sat and Sun's Sales distributions are unique. In dataset,

DayofWeek is represented as numeric number 1-7. From intuitive, we know that there is no linear relationship from 1-7 number to Sales data. We treat DayofWeek as four factors, Mon, Weekday(Tue-Fri), Sat, Sun.

```
#Separated Date column into year, month and day for prediction variables
library(lubridate)
data_year<-year(order_data$Date)
data_month<-month(order_data$Date)
data_day<-day(order_data$Date)
data_week<-week(order_data$Date)

#Combining the 3 different columns - month, day and year with the order_data data frame
combined_data<-cbind(order_data[1:3], data_year, data_month, data_day, data_week, order_data[4:18])

#Changing number of days in a week with the corresponding character names - sales on Monday, Saturday, Sunday and rest of the weekdays are different
combined_data$DayofWeek<-as.numeric(combined_data$DayofWeek)
combined_data$DayofWeek[combined_data$DayofWeek==1]<-"Mon"
combined_data$DayofWeek[combined_data$DayofWeek==2 | combined_data$DayofWeek==3 | combined_data$DayofWeek==4 | combined_data$DayofWeek==5 ]<-"Weekday"
combined_data$DayofWeek[combined_data$DayofWeek==6]<-"Sat"
combined_data$DayofWeek[combined_data$DayofWeek==7]<-"Sun"
combined_data$DayofWeek<-as.factor(combined_data$DayofWeek)
```

**Feature Engineering:** Combine Competition Distance, CompetitionOpenSince Month, CompetitionOpenSinceYear to a HaveCompetitor. The HaveCompetitor indicator indicates on a certain day whether a certain store has a competitor.

```
#Added a new column based on values in CompetitionOpenSinceMonth and CompetitionOpenSinceYear - FEATURE ENGINEERING
combined_data$HaveCompetitor<-ifelse((is.na(combined_data$CompetitionOpenSinceMonth) & is.na(combined_data$CompetitionOpenSinceYear)) & is.na(combined_data$CompetitionDistance)), 0, 1)
```

Cleaning on StateHoliday column in dataset.

Cleaning on CompetitionDistance column in dataset. In this column we replace NAs with CompetitionDistance as a large number 100000. This method enables us to only one CompetitionDistance feature. It also models the no competitor case by weakening CompetitionDistance impact.

```
#Merging b and c columns of StateHoliday because they have similar sales value as compared to others, a = public holiday, b = Easter holiday, c = Christmas, 0 = None
combined_data$StateHoliday<-as.character(combined_data$StateHoliday)
combined_data$StateHoliday[combined_data$StateHoliday=="b"|combined_data$StateHoliday=="c"]<-"d"
combined_data$StateHoliday<-as.factor(combined_data$StateHoliday)

#CompetitionDistance - replacing NAs with 100000 where HaveCompetitor = 0
combined_data$CompetitionDistance[(combined_data$HaveCompetitor==0)]<-100000
```

**Feature Engineering:** Combine Promo2, Promo2SinceWeek, Promo2SinceYear and Promointerval to a promo2 indicator in historical sales data. The indicator indicates on a certain day whether a certain store is on promotion 2.

```
#Combine Promo2, Promo2SinceWeek, Promo2SinceYear and Promointerval to a promotion 2 indicator in historical sales data. The indicator indicates on a certain day whether a certain store is
combined_data$Promo2Indicator<-ifelse(((combined_data$Promo2==0) & (is.na(combined_data$Promo2SinceYear)) & (is.na(combined_data$Promo2SinceWeek))
& (combined_data$PromoInterval=="")), 0, 1)
```

**Feature Engineering:** Adding a Promo2Month column

PromoInterval records the first month of each email marketing, replaced it with Promo2Month, which has number of months when the store conducted the most recent email promotion.

**Steps for adding Promo2month column:**

1. Extract each month name from PromoInterval column and treat them as promointerval quarters.

```
#PromoInterval records the first month of each email marketing, replaced it with Promo2Month, which has number of months when the store conducted the most recent email promotion
library(stringi)

# Extracting each month name from PromoInterval column and treat them as promointerval quarters
#Fourth Quarter
w<-as.character(combined_data$PromoInterval)
promo_month<-function(w,n){substr(w,nchar(w)-n+1,nchar(w))}
combined_data$QuarterFour<-promo_month(w,3)

#Third Quarter
x<-as.character(combined_data$PromoInterval)
promo_month3<-function(x,n){substr(x,nchar(x)-n+1,nchar(x))}
combined_data$QuarterThree<-promo_month3(x,7)
combined_data$QuarterThree<-stri_sub(combined_data$QuarterThree,1,3)

#Second Quarter
y<-as.character(combined_data$PromoInterval)
promo_month2<-function(y,n){substr(y,nchar(y)-n+1,nchar(y))}
combined_data$QuarterTwo<-promo_month2(y,11)
combined_data$QuarterTwo<-stri_sub(combined_data$QuarterTwo,1,3)

#First Quarter
z<-as.character(combined_data$PromoInterval)
promo_month1<-function(z,n){substr(z,nchar(z)-n+1,nchar(z))}
combined_data$QuarterOne<-promo_month1(z,15)
combined_data$QuarterOne<-stri_sub(combined_data$QuarterOne,1,3)
```



2. Changing the month names in corresponding columns to numeric values as per the months in a year.

```
#Changing the month names in corresponding columns to numeric values as per the months in a year
combined_data$QuarterFour[which(combined_data$QuarterFour=="Oct")]<-"10"
combined_data$QuarterFour[which(combined_data$QuarterFour=="Nov")]<-"11"
combined_data$QuarterFour[which(combined_data$QuarterFour=="Dec")]<-"12"
combined_data$QuarterThree[which(combined_data$QuarterThree=="Jul")]<-"7"
combined_data$QuarterThree[which(combined_data$QuarterThree=="Aug")]<-"8"
combined_data$QuarterThree[which(combined_data$QuarterThree=="Sep")]<-"9"
combined_data$QuarterTwo[which(combined_data$QuarterTwo=="Apr")]<-"4"
combined_data$QuarterTwo[which(combined_data$QuarterTwo=="May")]<-"5"
combined_data$QuarterTwo[which(combined_data$QuarterTwo=="Jun")]<-"6"
combined_data$QuarterOne[which(combined_data$QuarterOne=="Jan")]<-"1"
combined_data$QuarterOne[which(combined_data$QuarterOne=="Feb")]<-"2"
combined_data$QuarterOne[which(combined_data$QuarterOne=="Mar")]<-"3"
```

3. Converting the four quarter columns to numeric value columns.

```
#Converting the four quarter columns to numeric value columns
combined_data$QuarterFour<-as.numeric(combined_data$QuarterFour)
combined_data$QuarterThree<-as.numeric(combined_data$QuarterThree)
combined_data$QuarterTwo<-as.numeric(combined_data$QuarterTwo)
combined_data$QuarterOne<-as.numeric(combined_data$QuarterOne)
```

4. Calculating the number of months by applying the following formula to data\_year, data\_month, Promo2SinceYear and QuarterFour columns

```
#Calculating the number of months by applying the following formula to data_year, data_month, Promo2SinceYear and QuarterFour columns
combined_data$Promo2SinceYear<-as.numeric(combined_data$Promo2SinceYear)
b<-(12 * (combined_data$data_year - combined_data$Promo2SinceYear)) + (combined_data$data_month - combined_data$QuarterFour)
c<-(12 * (combined_data$data_year - combined_data$Promo2SinceYear)) + (combined_data$data_month - combined_data$QuarterThree)
d<-(12 * (combined_data$data_year - combined_data$Promo2SinceYear)) + (combined_data$data_month - combined_data$QuarterTwo)
e<-(12 * (combined_data$data_year - combined_data$Promo2SinceYear)) + (combined_data$data_month - combined_data$QuarterOne)
```

5. Running the for-loop for taking out the positive number of months for the duration when the store performed most recent promotion

Taking out negative values (starting promotion before shop opened)

```
#Running the for-loop for taking out the positive number of months for the duration when the store performed most recent promotion
#Taking out negative values (starting promotion before shop opened)
combined_data$PromoMonth<-ifelse(combined_data$Promo2SinceYear>combined_data$data_year,0,
  ifelse(combined_data$QuarterFour<=combined_data$data_month, b,
    ifelse(combined_data$QuarterThree<=combined_data$data_month, c,
      ifelse(combined_data$QuarterTwo<=combined_data$data_month, d,
        ifelse(combined_data$QuarterOne<=combined_data$data_month, e,
          0))))))
```

## 6. Replacing the NAs with zero in the PromoMonth Column

```
#Replacing the NAs with zero in the PromoMonth Column  
combined_data$PromoMonth[is.na(combined_data$PromoMonth)]<-0
```

### Feature Engineering: Creating new column Expected\_Sales.

```
#Taking the mean of Sales  
mean<-mean(combined_data$Sales)  
  
#Creating new column Expected_Sales  
combined_data$Expected_Sales<-ifelse(combined_data$Sales<mean,0,1)  
combined_data$Expected_Sales<-factor(combined_data$Expected_Sales, levels=c(0,1), labels=c("Below Average", "Above Average"))
```

Write data to csv file named **cleanedData\_RossmannSales.csv** file.

## 1.8 Regression Models

### 1.8.1 Overall Design

Read the Rossmann Sales Data.

Cleanse the data as explained in Data Cleansing section.

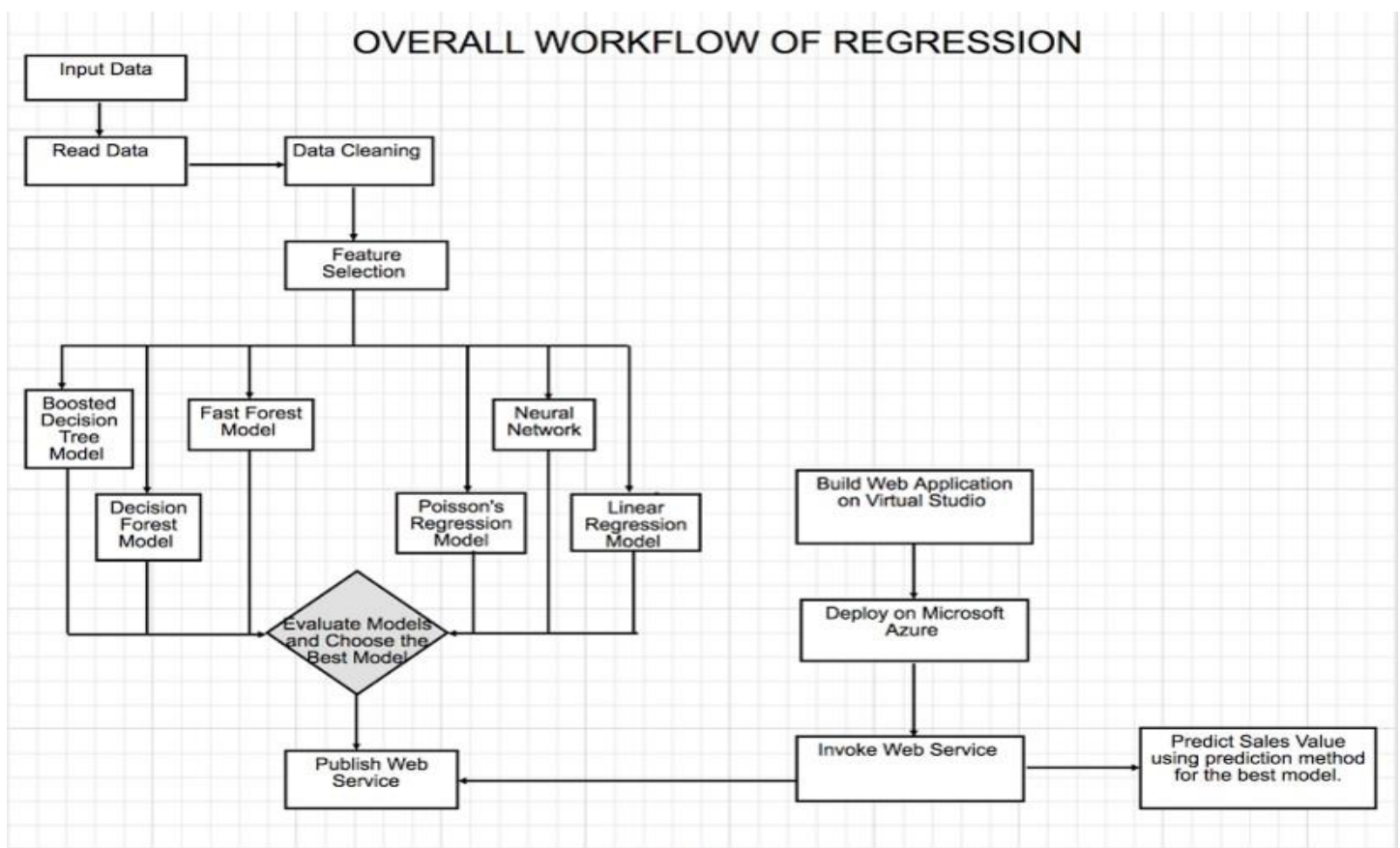
Implement various regression models.

Compare the models through MAE (mean absolute error) and RMSPE (Root Mean Square Percentage Error) and choose the best model.

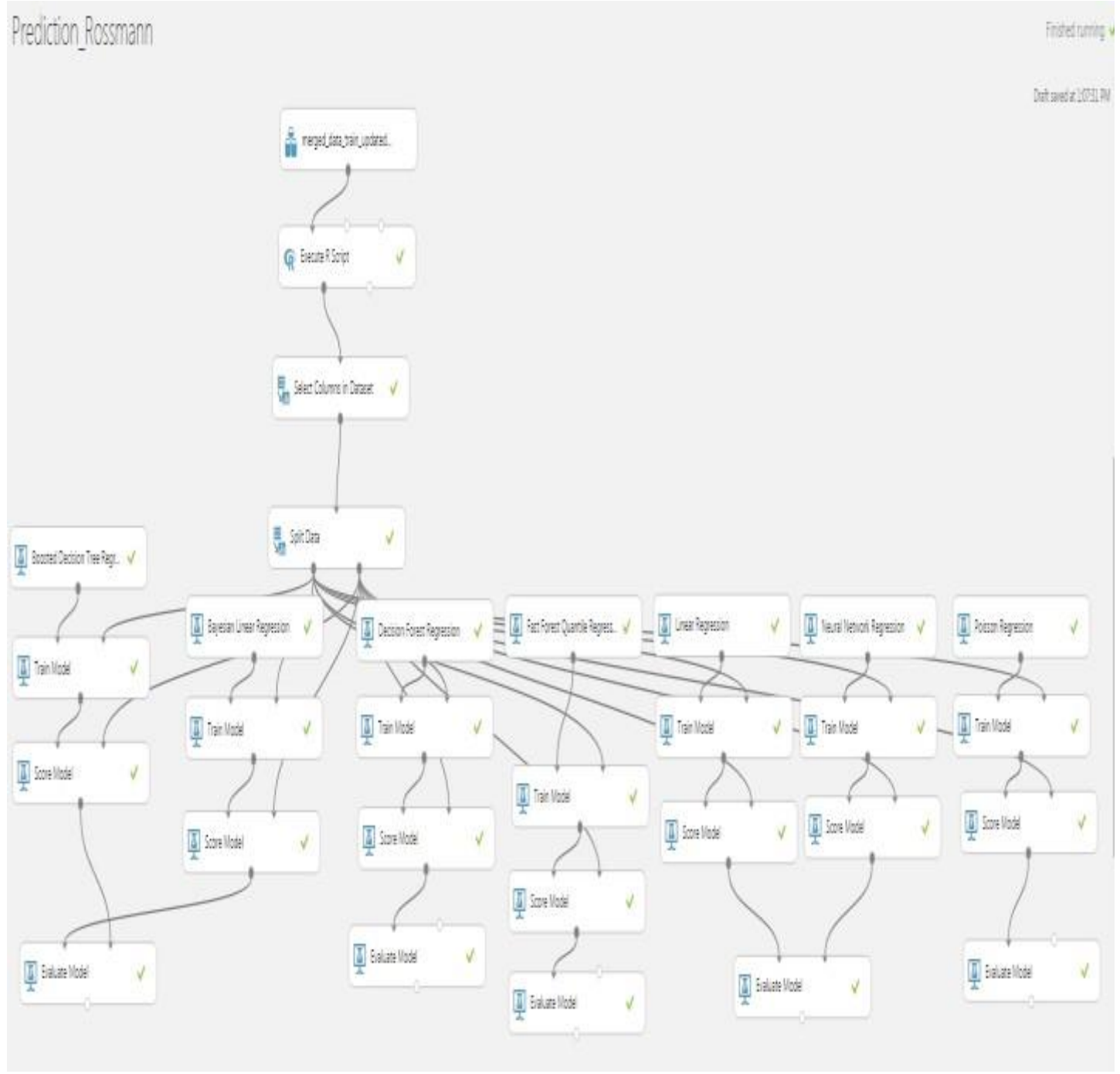
Deploy the best regression model as a web service.

Build web application using visual studio and deploy it on Microsoft Azure.

Predict value of Sales through this web application.



## 1.8.2 Azure Models



### 1.8.3 Boosted Decision Tree Regression



### 1.8.4 Bayesian Linear Regression

Prediction\_Rossmann > Evaluate Model > Evaluation results

rows	columns						
1	6						
		Negative Log Likelihood	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
view as							
		1978584.652651	2001.524187	2742.253449	0.8728	0.775235	0.224765





## 1.8.5 Decision Forest Regression

Prediction\_Rossmann > Evaluate Model > Evaluation results

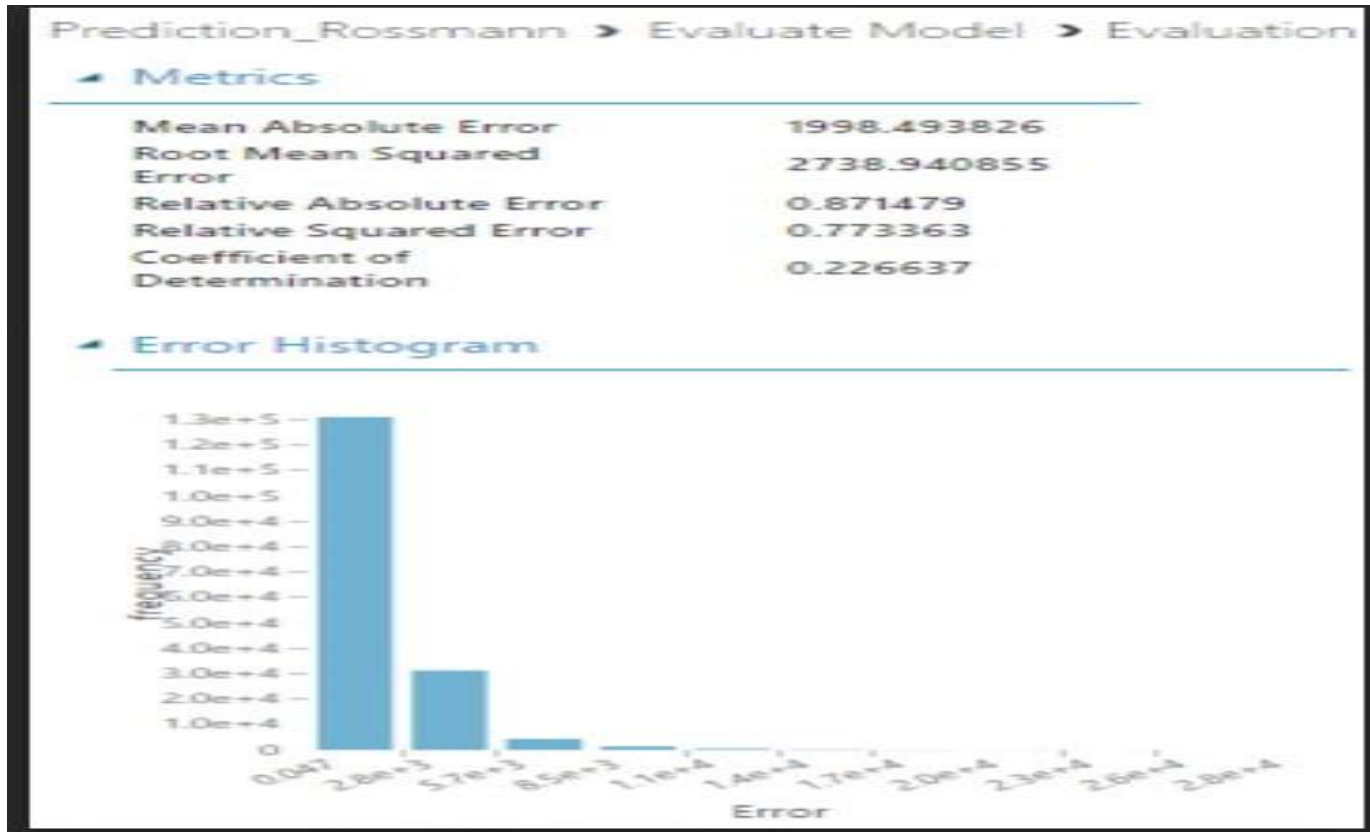
rows  
1

columns  
6

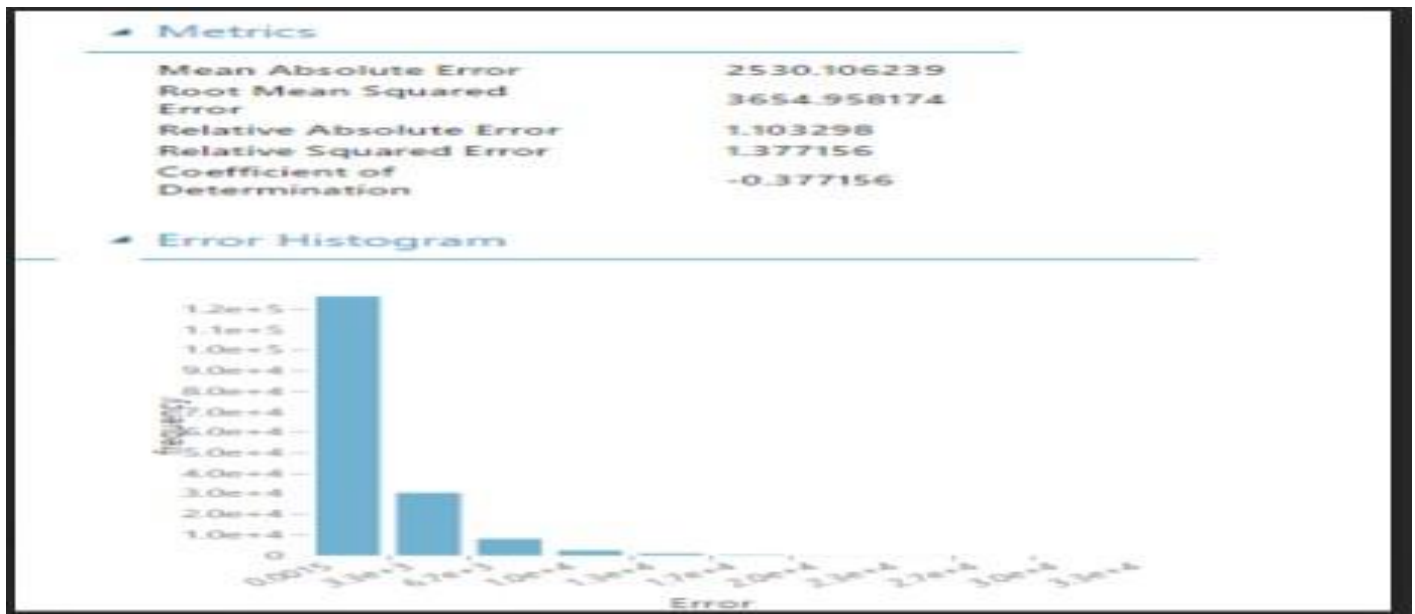
view as  
 

Negative Log Likelihood	Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
1406513.829959	698.657323	1070.846979	0.304662	0.118215	0.881785

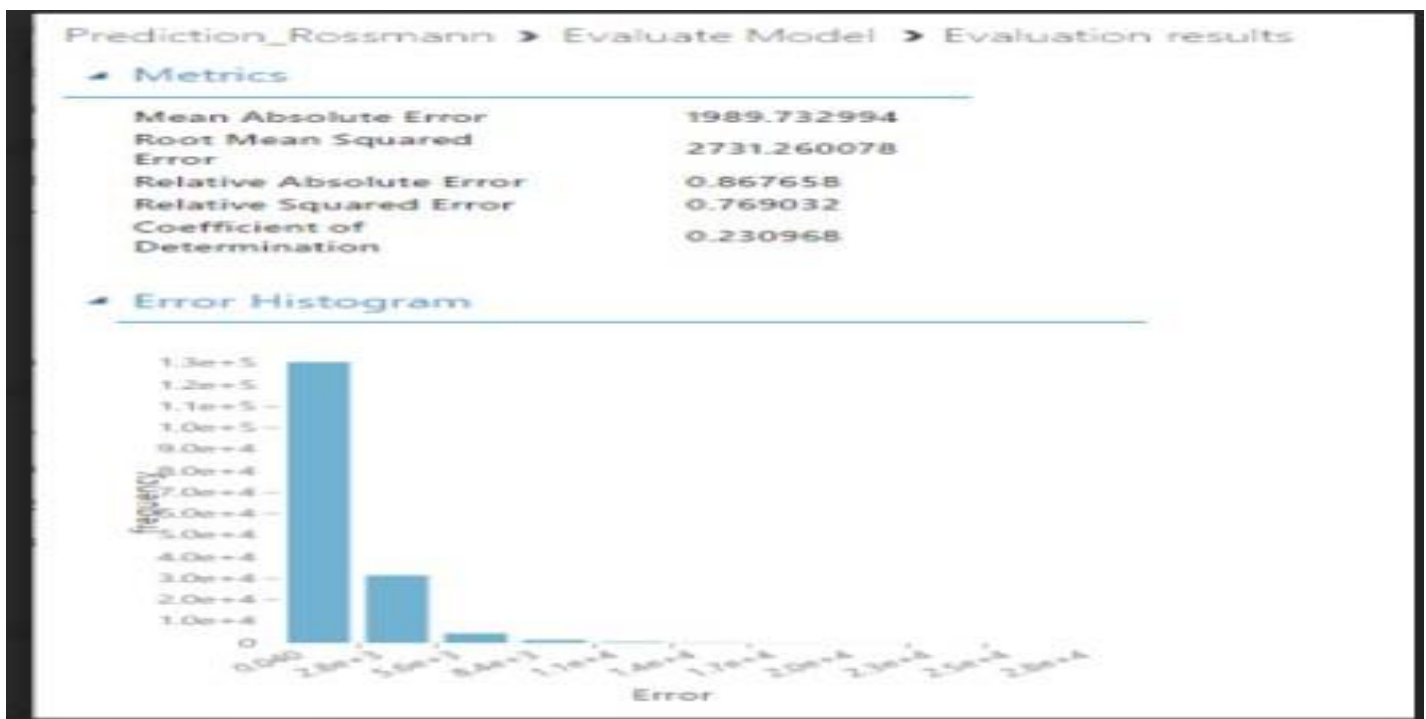
## 1.8.6 Linear Regression



## 1.8.5 Neural Network Regression



## 1.8.8 Poisson Regression



### 1.8.9 ARIMA Model



ARIMA model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied to reduce the non-stationarity.<sup>[1]</sup>

The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of each of these features is to make the model fit the data as well as possible.

Arima Model ▶ Execute R Script ▶ Result Dataset

rows  
1

columns  
6

view as  
 

ME	RMSE	MAE	MPE	MAPE	MASE
32.258469	1010.457704	783.658649	-3.534265	16.792223	0.989129

## 1.8.10 Summary

Model Name	MAE	RMSE
Boosted Decision Tree Regression	1,224	1,662 (Approx. RMSPE = 14.64%)
Bayesian Linear Regression	2,001	2,742 (Approx. RMSPE = 23.58%)
Decision Forest Regression	698	1,070 (Approx. RMSPE = 11.74%)
Linear Regression	1,998	2,738 (Approx. RMSPE = 23.16%)
Neural Network Regression	2,530	3,654 (Approx. RMSPE = 34.93%)
Poisson Regression	1,989	2,731 (Approx. RMSPE = 26.89%)
ARIMA Model	7,83	1,010 (Approx. RMSPE = 10.55%)

## 1.8.11 Conclusion

Out of all these models we chose Decision Forest Regression Model because it had minimum MAPE i.e. **698** and minimum RMSE i.e. **1,070**. It had the best combination amongst all the other models.

## 1.8.12 Web Service

Below are the steps to create a web service of Regression model:

- Set-up web service

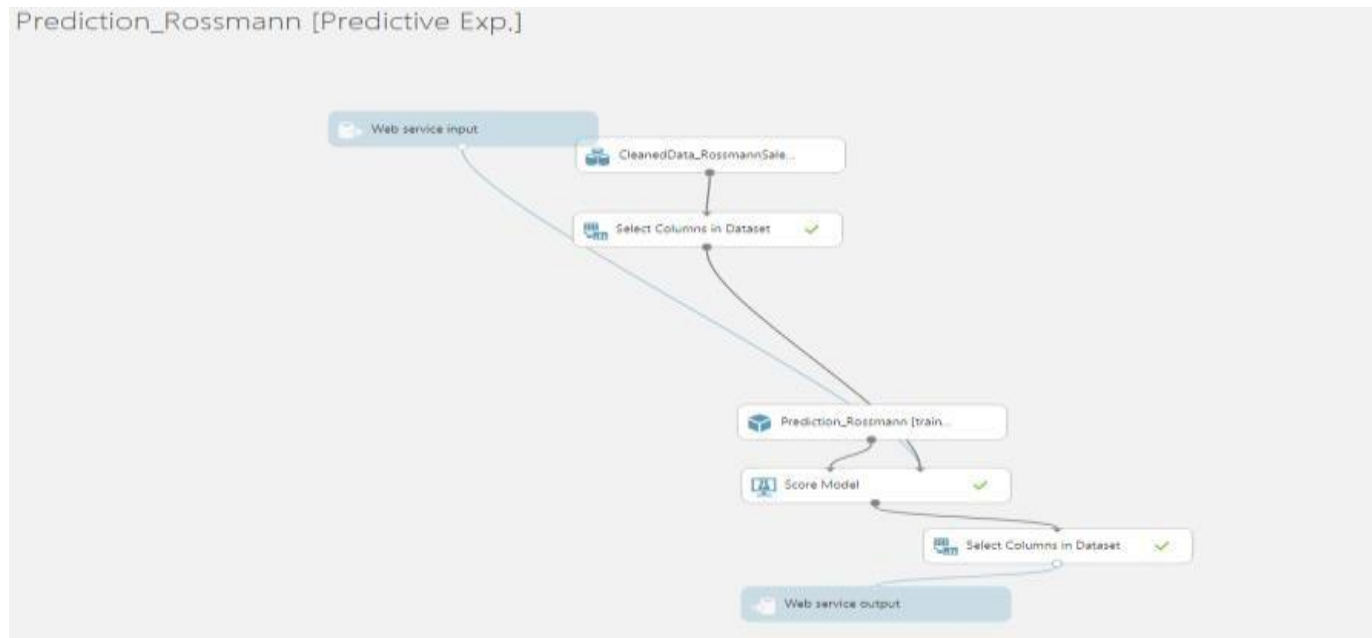
- Save Decision Forest Regression Model's train model as a Trained Model.

- Modules that were used for training are removed. Specifically:

  - Decision Forest Regression Model

  - Train Model

  - Split Data



Then Web Service input and Web Service output is added.  
Service is deployed.

#### prediction\_rossmann [predictive exp.]

DASHBOARD CONFIGURATION

General

Published experiment

View snapshot View latest

Description

No description provided for this web service.

API key

@v8DQ+LqMhYXqqrNypFWvREHWskicEXUk4ZDuhkdyDajfakthMsalQxWHTmdSX8ON8LgXyku8NuW71Q==

Default endpoint

API HELP PAGE

REQUEST/RESPONSE

BATCH EXECUTION

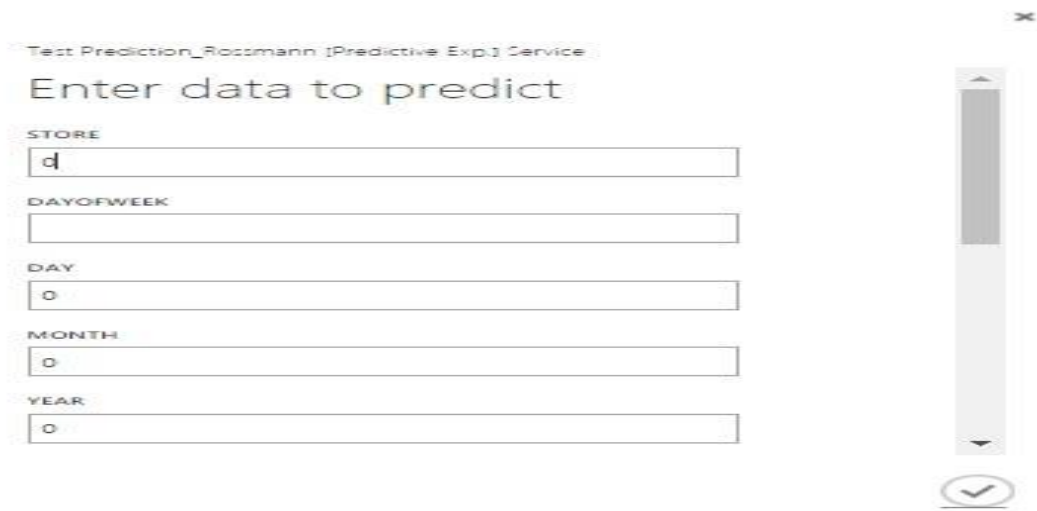
TEST

Test

APPS

Excel 2013 or later (1) Excel 2010 or earlier workbook

Excel 2013 or later workbook



## 1.9 Classification Models

### 1.9.1 Overall Design

Read the Rossmann Sales Data.

Cleanse the data as explained in Data Cleansing section.

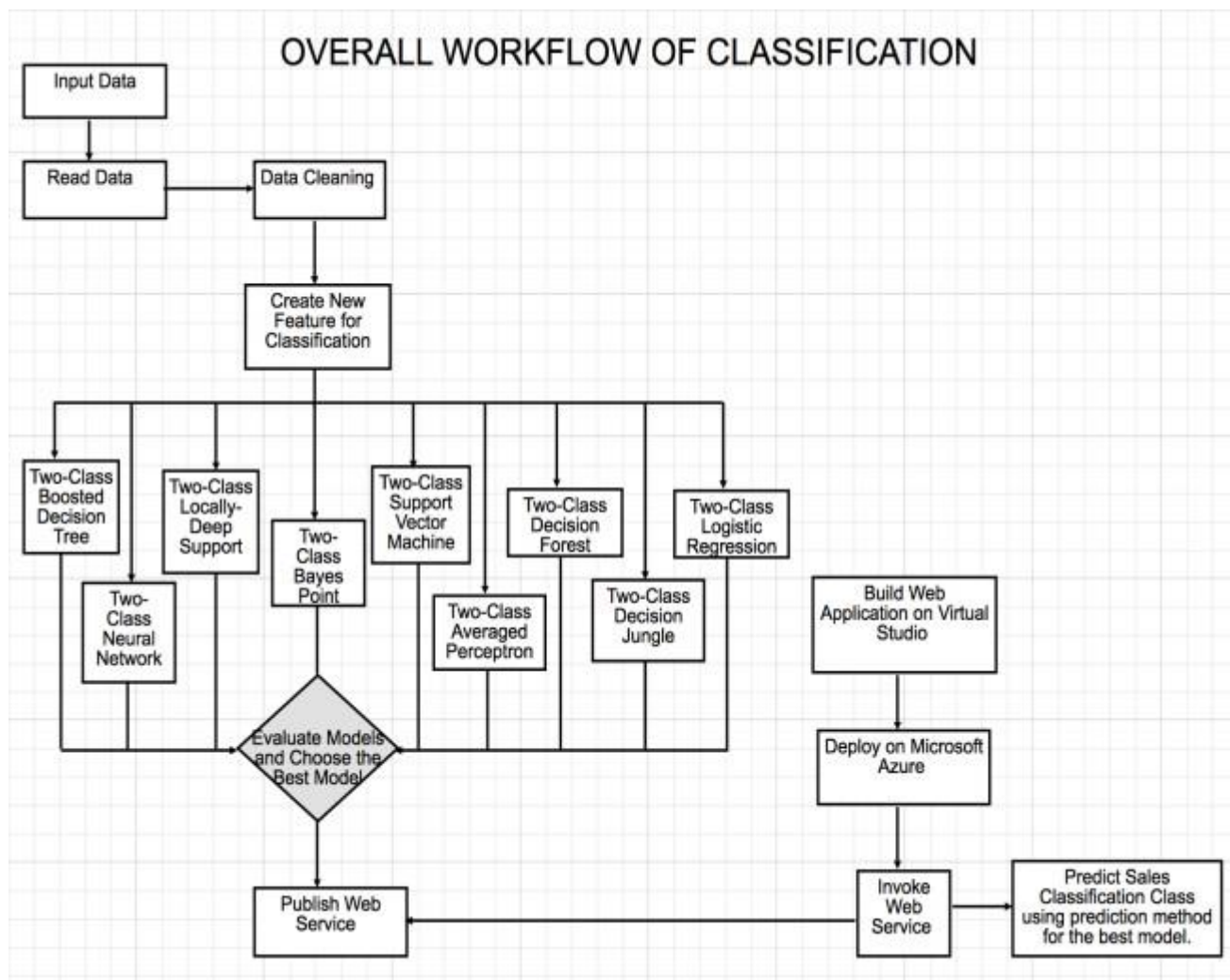
Implement various classification models.

Compare the models through Error Percentage, Accuracy and Precision to choose the best model.

Deploy the best classification model as a web service.

Build web application using visual studio and deploy it on Microsoft Azure.

Predict the class of predicted sales value through this web application.



Classification\_Rossmann

Finished running

merged data train updated

Evaluate F1 Score

Select Columns in Dataset

Split Data

Two Class Support Vector

Two Class Averaged Perceptron

Two Class Bayes Point Machine

Two Class Decision Forest

Two Class Decision Jungle

Two Class Logistic Regression

Two Class Locally Biased SVM

Two Class Neural Network

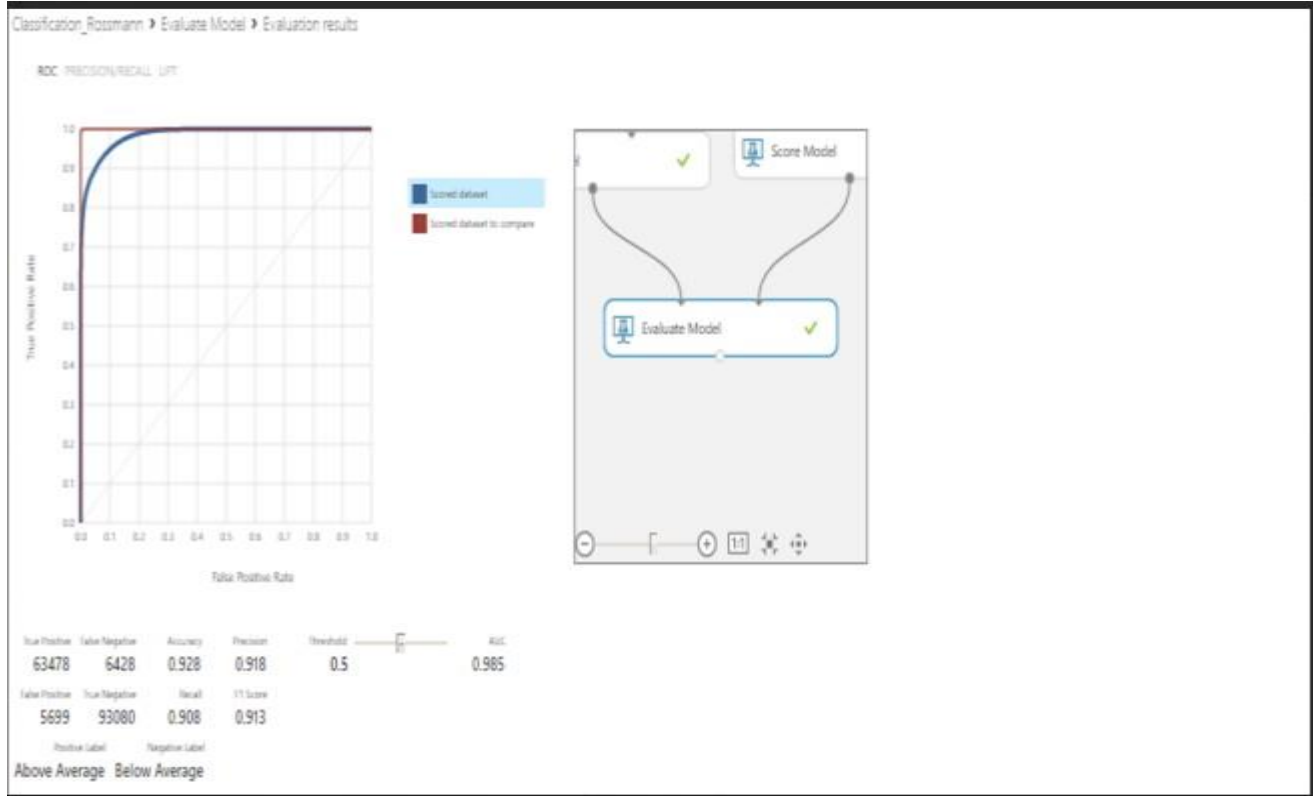
Train Model

Score Model

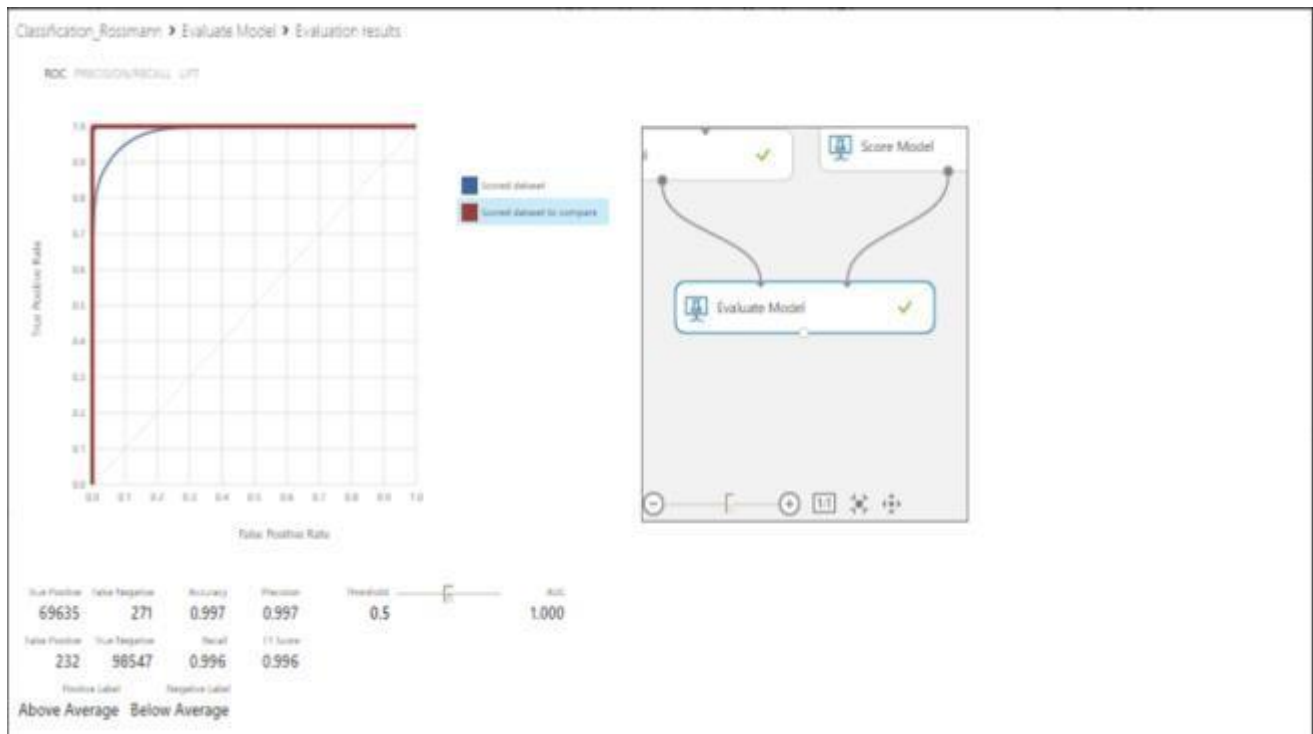
Evaluate Model



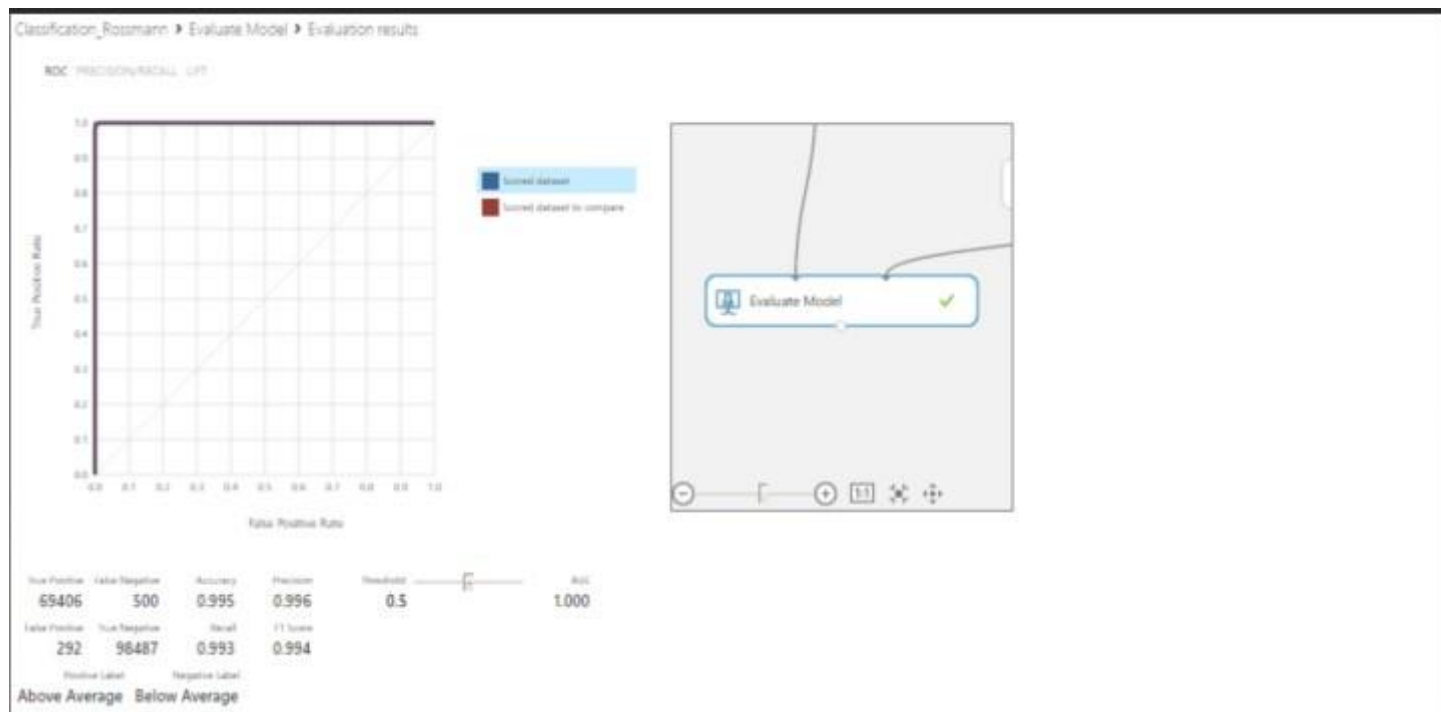
### 1.9.3 Two-Class Support Vector Machine



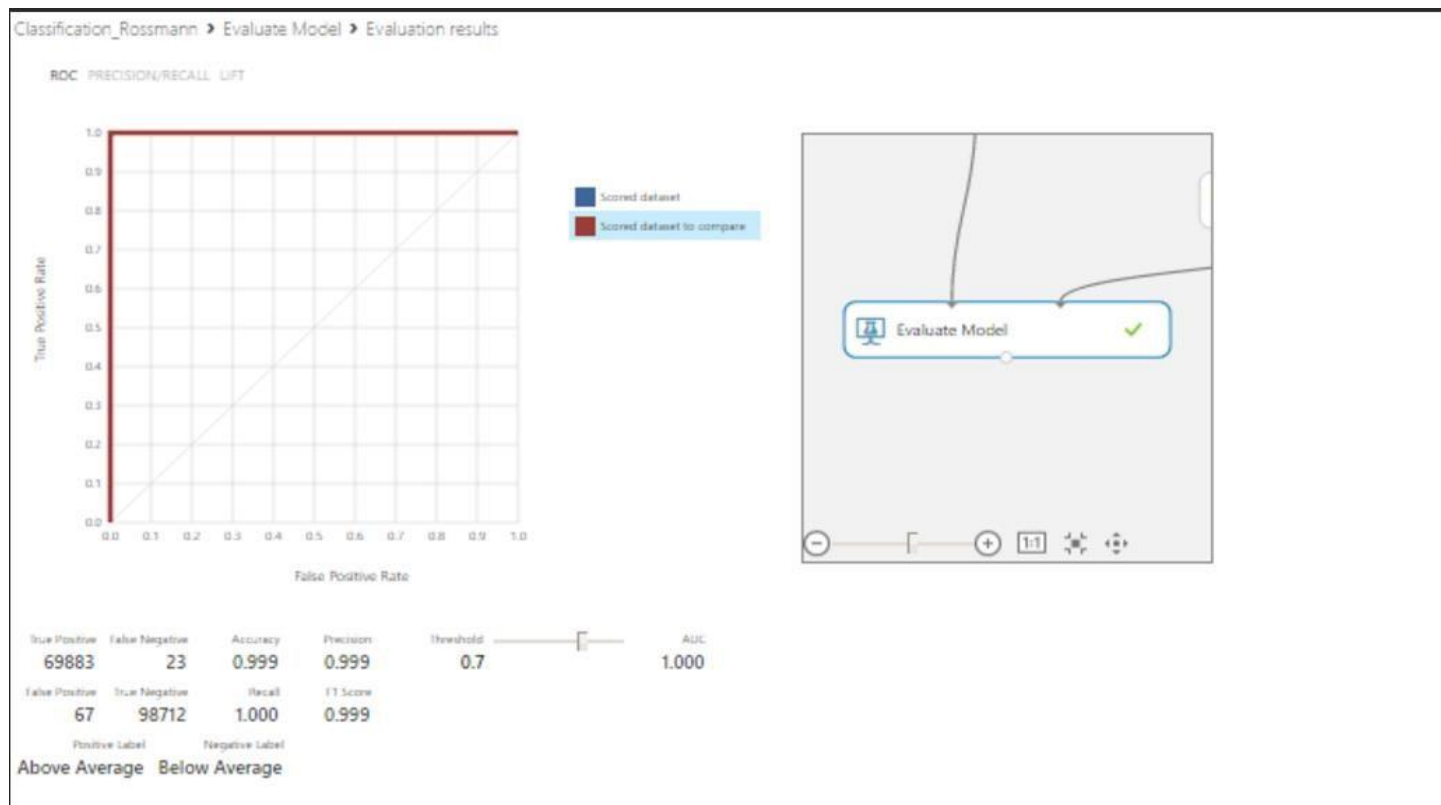
### 1.9.4 Two-Class Averaged Perceptron



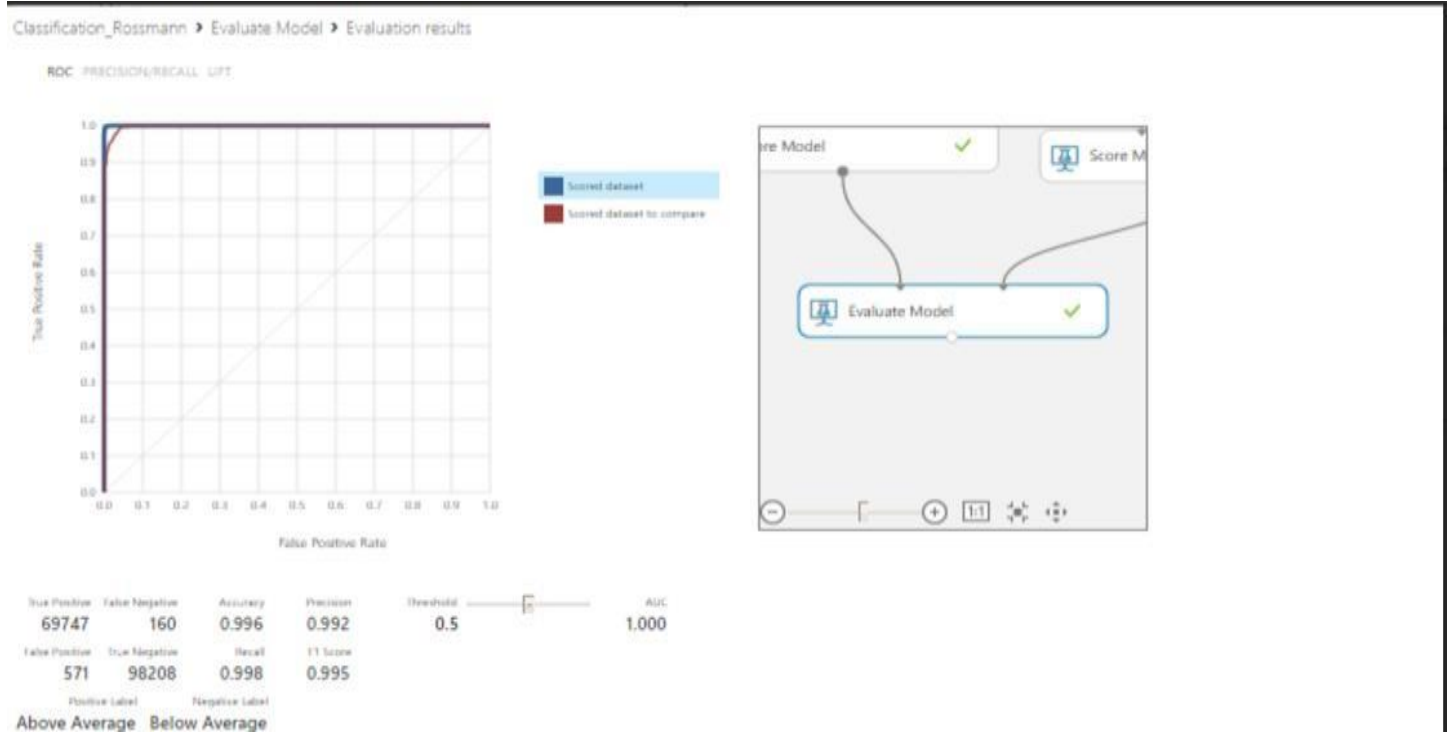
## 1.9.5 Two-Class Bayes Point Machine



## 1.9.6 Two-Class Boosted Decision Tree



## 1.9.7 Two-Class Decision Forest



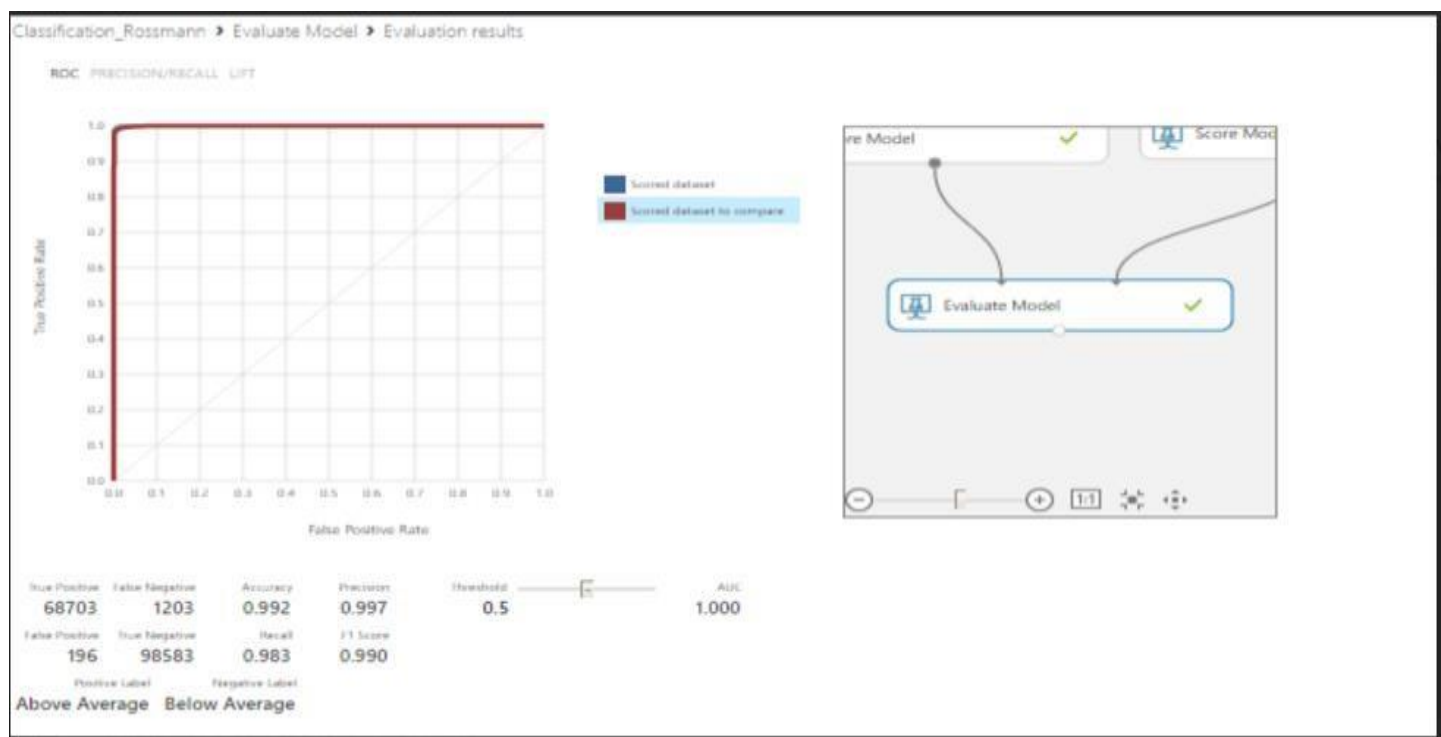
## 1.9.8 Two-Class Decision Jungle



## 1.9.9 Two-Class Logistic Regression



## 1.9.10 Two-Class Locally-Deep Support Vector Machine



## 1.9.11 Two Class Neural Network



## 1.9.12 Summary

Model Name	Accuracy	Precision	Error Percentage
Two Class Support Vector Machine	0.928	0.918	0.71
Two Class Averaged Perceptron	0.997	0.997	0.3
Two Class Bayes Point Machine	0.995	0.996	0.4
Two Class Boosted Decision Tree	0.999	0.999	0.05
Two Class Decision Forest	0.996	0.992	0.43
Two Class Decision Jungle	0.973	0.945	2.6
Two Class Logistic Regression	0.994	0.996	0.5

<b>Two Class Locally Deep SVM</b>	<b>0.992</b>	<b>0.997</b>	<b>0.8</b>
<b>Two Class Neural Network</b>	<b>0.986</b>	<b>0.969</b>	<b>1.3</b>

### 1.9.13 Conclusion

So we chose Two-Class Boosted Decision Tree classification model over all other models as it has the best accuracy and precision and less error percentage.

### 1.9.14 Web Service

Below are the steps to create a web service of classification model:

- Set-up web service

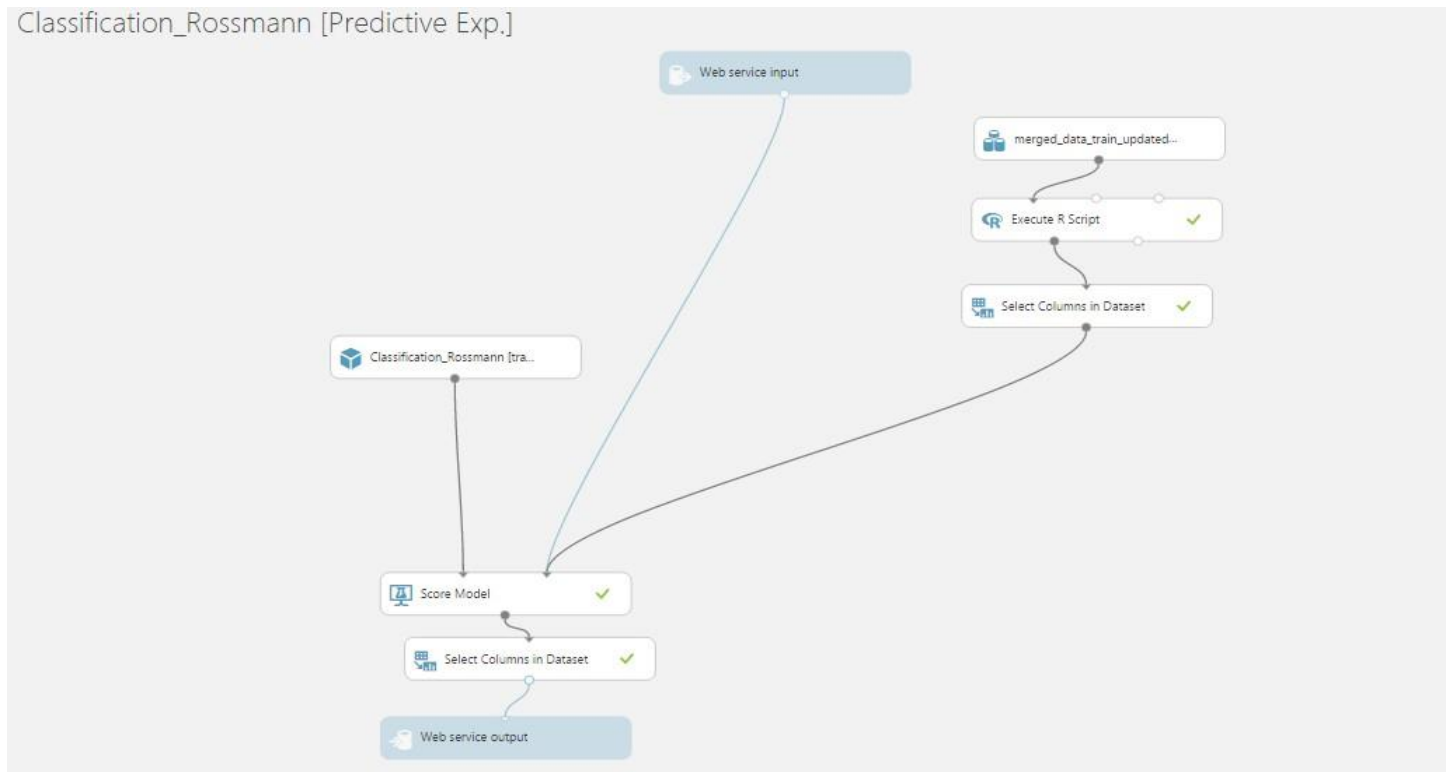
- Save Two-Class Boosted Decision Tree's train model as a Trained Model.

- Modules that were used for training are removed. Specifically:

  - Boosted Decision Tree Model

  - Train Model

  - Split Data



Then Web Service input and Web Service output is added.  
Service is deployed.

### classification\_rossmann [predictive exp.]

DASHBOARD CONFIGURATION

General

Published experiment

[View snapshot](#) [View latest](#)

Description

No description provided for this web service.

API key

/PNolgpouVMHICfMm2D7uHN+1gFIcLnveGL54DdA8CImGvH7LnWv8v75t5+QGJaMKn/3VdlROsVSw77QyphQw=

Default Endpoint

API HELP PAGE

TEST

APPS

REQUEST/RESPONSE

Test

Excel 2013 or later Excel 2010 or earlier workbook

BATCH EXECUTION

Excel 2013 or later workbook

×

Test Classification\_Rossmann [Predictive Exp.] Service

Enter data to predict

STORE

d

DAYOFWEEK

Mon

DAY

0

MONTH

0

YEAR

0

↑

↓

✓



## 1.10 Clustering

Clustering can be considered the most important ***unsupervised learning*** problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data.

A loose definition of clustering could be ‘the process of organizing objects into groups whose members are similar in some way’.

A ***cluster*** is therefore a collection of objects which are ‘similar’ between them and are ‘dissimilar’ to the objects belonging to other clusters.

### 1.10.1 The Algorithm: K-Means Clustering

K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problems. The algorithm follows a simple and easy way to classify a given data set through a certain number of clusters (assume  $k$  clusters) fixed a priori. The main idea is to define  $k$  centroids, one for each cluster. This algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function

$$J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$$

where  $\|x_i^{(j)} - c_j\|^2$  is a chosen distance measure between a data point  $x_i^{(j)}$  and the cluster centre  $c_j$ , is an indicator of the distance of the  $n$  data points from their respective cluster centres.

### 1.10.2 Web Service

Below are the steps to create a web service of classification model:

- Set-up web service

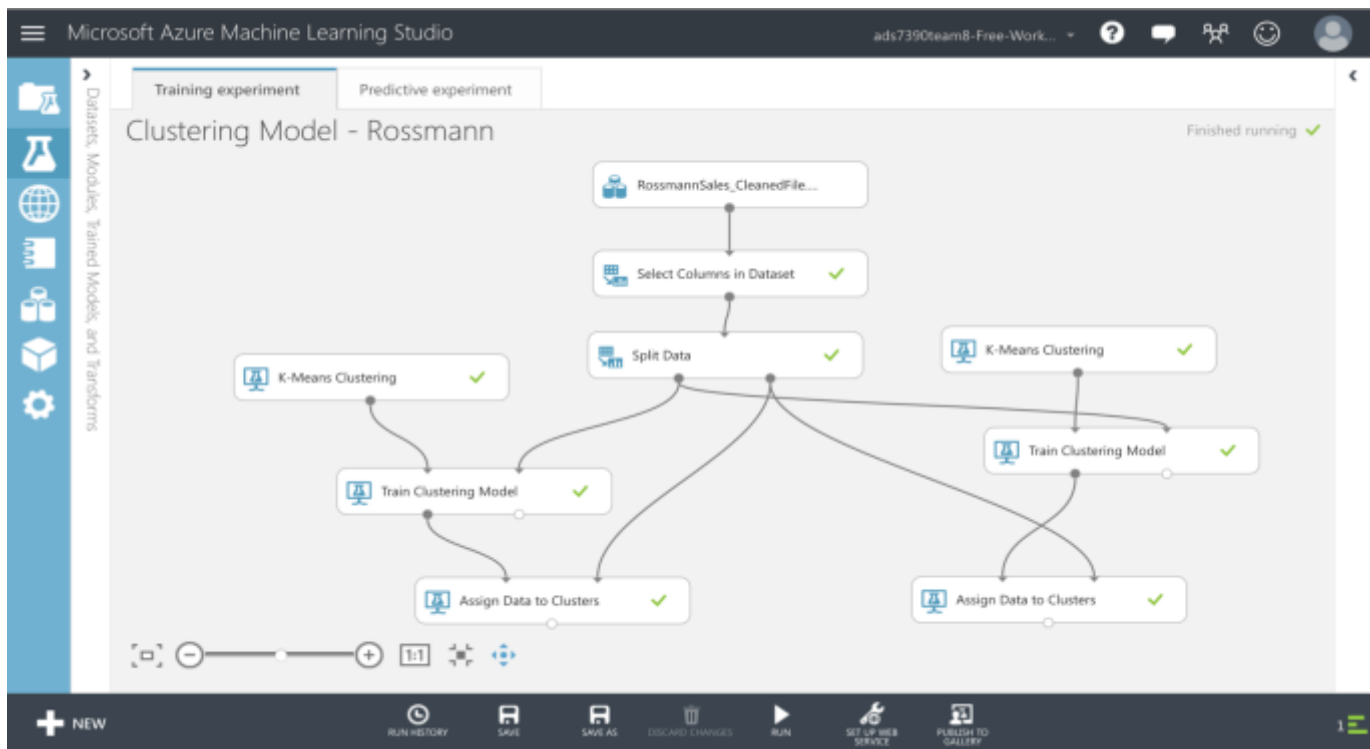
- Add dataset to be used for training clustering model.

Modules that were used for training are removed. Specifically:

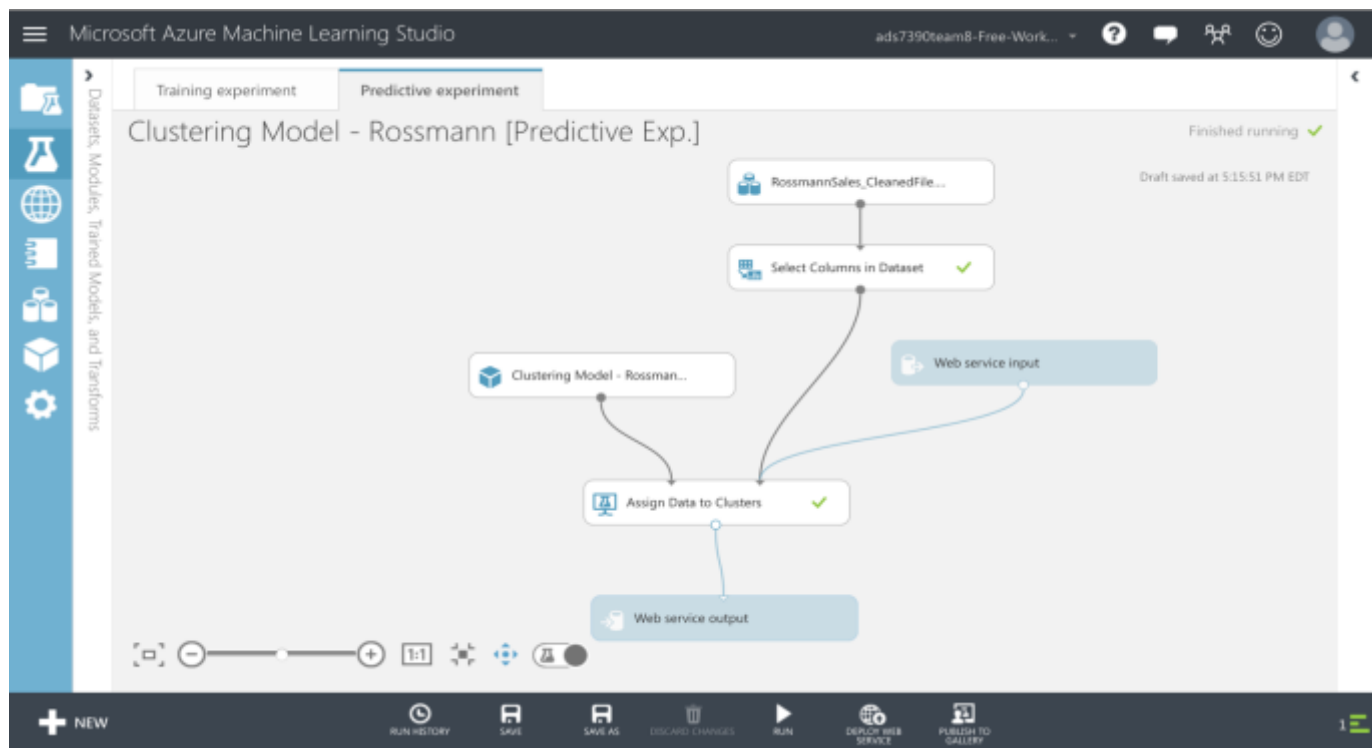
- K-Means Clustering
- Train Clustering Model
- Split
- Assign to Clusters

Experiments -

Training Experiment:



## Predictive Experiment:

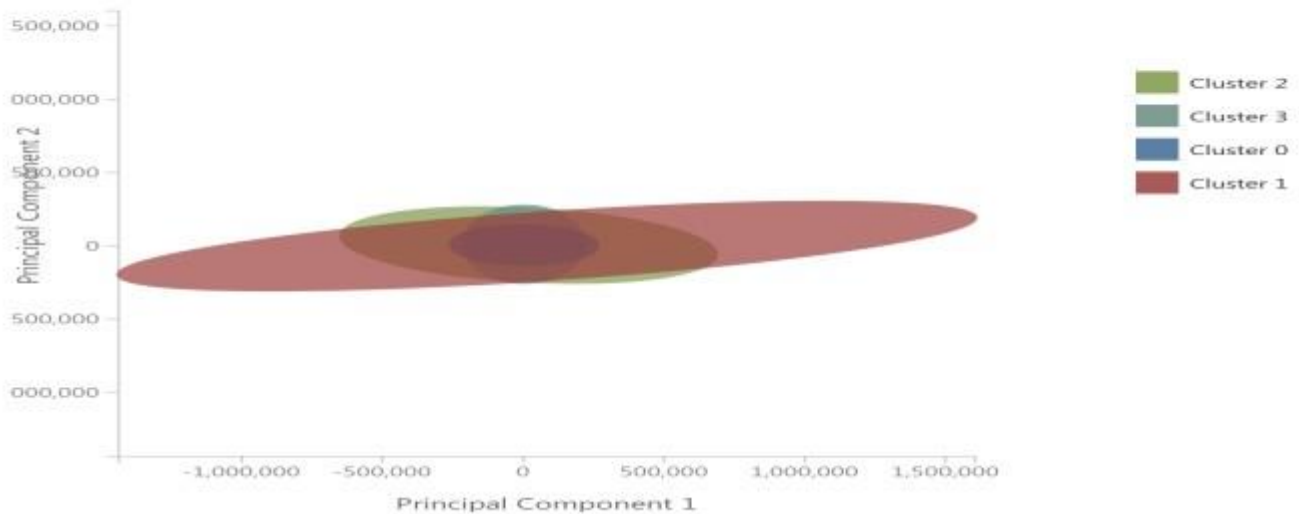


## 1.10.2 Cluster Formation

**K = 4**

The following is the clustering graph which depicts the K-Means Clustering for 17 variables in the cleaned train dataset. Here the **K = 4** called as centroids. The clusters show the overall similarity between the values.

Clustering Model - Rossmann > Train Clustering Model > Results dataset



The following is the dataset obtained by training the clustering model with Rossmann sales dataset with K=4. The 'Assignments' column designates which cluster does the value of the particular row belongs to.

Choosing Model > Random Forest > K-Means Clustering > Histogram

columns

4/22/2019 8:27

customer	CompetitionDistance	PromotedIndicator	Sales	ExpectedSales	Assignments	Distance to Cluster Center no.0	Distance to Cluster Center no.1	Distance to Cluster Center no.2	Distance to Cluster Center no.3
10430	5	10067	Alcove	0	14523.95862	88767.85187	5422.296268	14461.862167	
1498	5	8829	Alcove	0	9280.79461	81708.148276	10706.468424	1812.216225	
2768	5	11187	Alcove	0	4690.318216	82487.728439	17159.536788	1488.766851	
598	0	5053	Bedroom, Alcove	0	2484.043581	84932.048734	19402.000602	5838.770882	
5522	5	3990	Bedroom, Alcove	0	1414.741961	85771.608340	16411.841811	4831.867489	
1438	0	7980	Alcove	0	2726.82057	84877.66851	18073.278012	5852.816771	
10150	0	5214	Bedroom, Alcove	0	7330.962295	54916.47645	9708.848579	5961.060586	
19840	5	5275	Bedroom, Alcove	0	18688.020184	40491.103110	16827.27084	18886.050057	
2232	5	8184	Alcove	0	5246.970877	80999.681571	17823.112887	1381.218208	
818	5	8223	Alcove	0	3778.812048	84941.220526	19448.997184	1999.0652	
3899	5	4349	Bedroom, Alcove	0	1017.398225	85120.653518	10275.400861	7188.205871	
198	0	14808	Alcove	0	9708.073027	89340.5443	21351.106411	4300.074956	
1435	0	7200	Alcove	0	2676.784878	84880.518208	18588.897458	4698.580002	
5830	5	8128	Alcove	0	1013.781751	85085.124740	16578.47682	1447.784214	
3040	0	5705	Bedroom, Alcove	0	787.086812	85389.433480	17043.736084	5137.501378	
8784	5	7535	Alcove	0	1080.154182	74840.378821	12751.54444	7257.511748	
7130	5	5007	Bedroom, Alcove	0	1218.505184	84918.40094	10060.494402	8570.958462	
120020	5	5019	Bedroom, Alcove	1	18689.821071	14951.452182	80387.327045	86121.24481	
8850	5	5024	Bedroom, Alcove	0	3888.30942	78233.775427	13172.897778	7802.85328	
4880	5	3904	Bedroom, Alcove	0	1171.8118	77263.623524	12444.70490	5411.401369	
3810	0	4709	Bedroom, Alcove	0	1458.238854	81072.70844	18263.884843	4984.270584	
1070	0	4913	Bedroom, Alcove	0	3578.782712	84226.887119	18918.247321	6429.217288	
26480	5	4918	Bedroom, Alcove	0	11805.912238	18413.26516	8865.891084	23784.489812	
5882	0	6189	Bedroom, Alcove	0	9342.527797	76158.448776	14286.121189	6276.167588	

Statistics

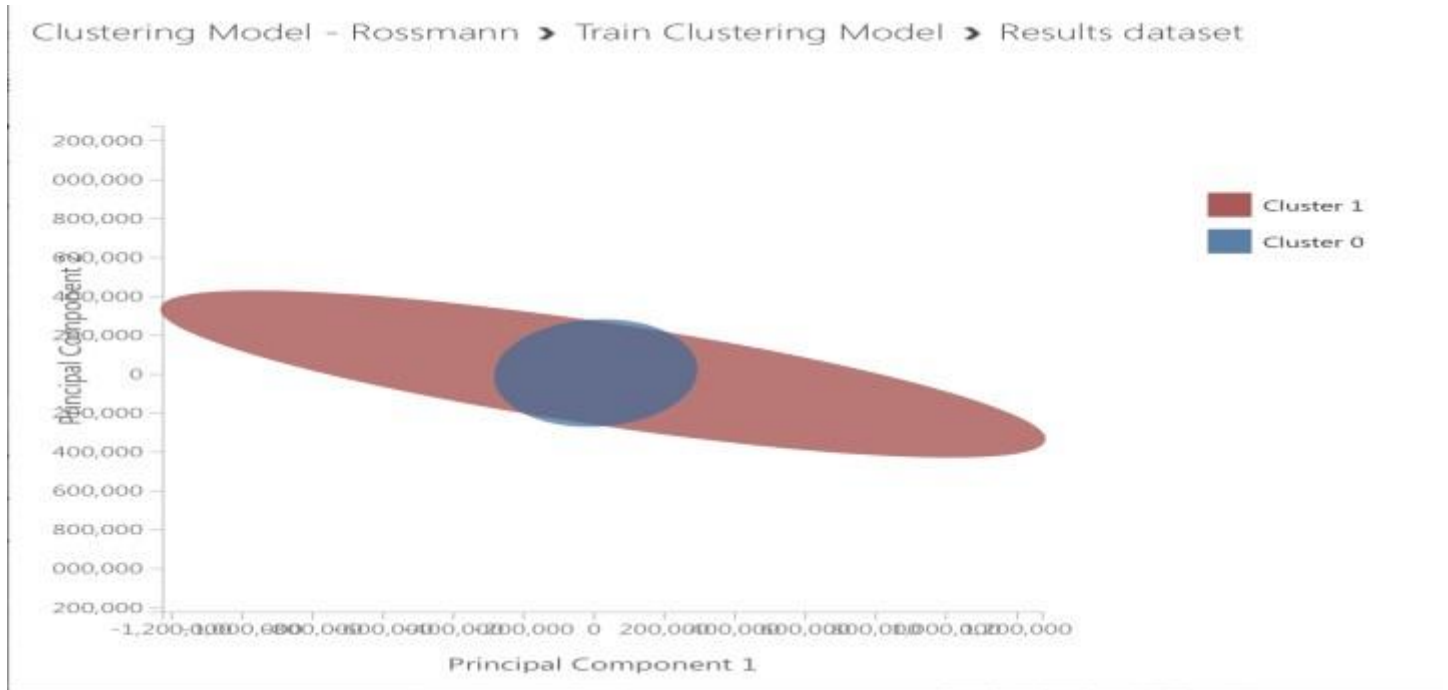
Unique Values: 4  
Missing Values: 0  
Feature Type: Categorical Feature

Assignments: 100%

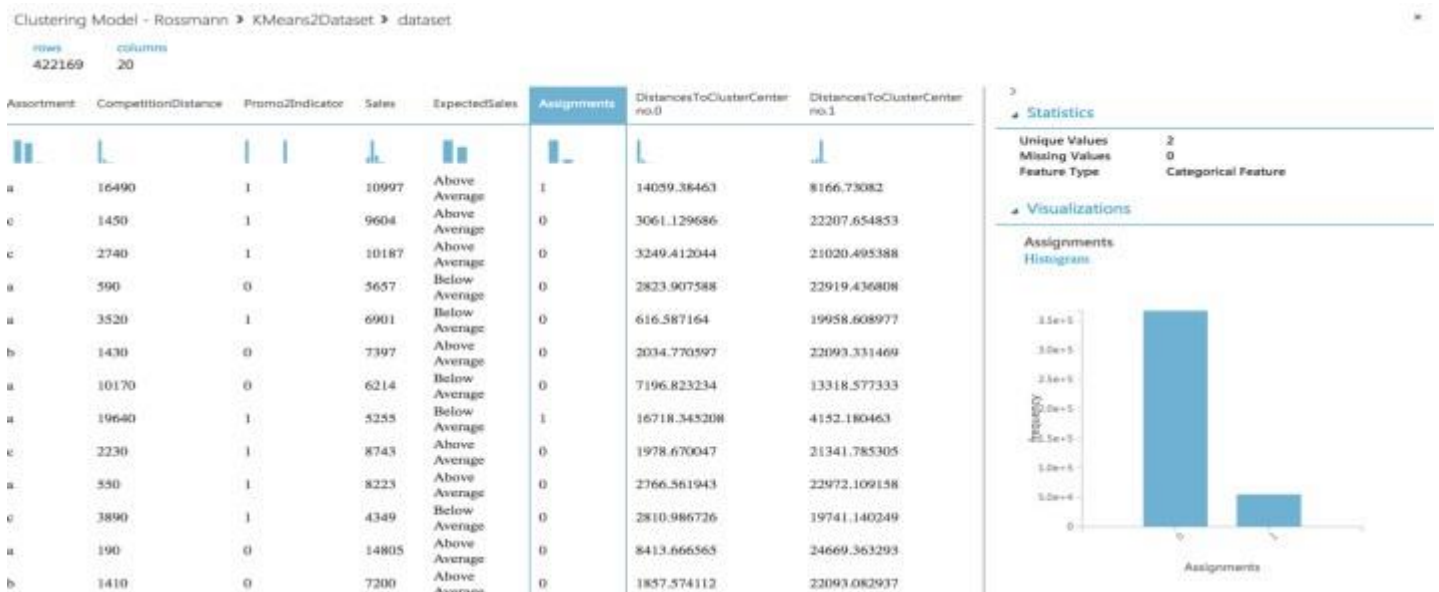
Cluster	Assignments
0	100%
1	0%
2	0%
3	0%

## K = 2

The following is the clustering graph which depicts the K-Means Clustering for 17 variables in the cleaned train dataset. Here the  $K = 2$  called as centroids. The clusters show the overall similarity between the values.

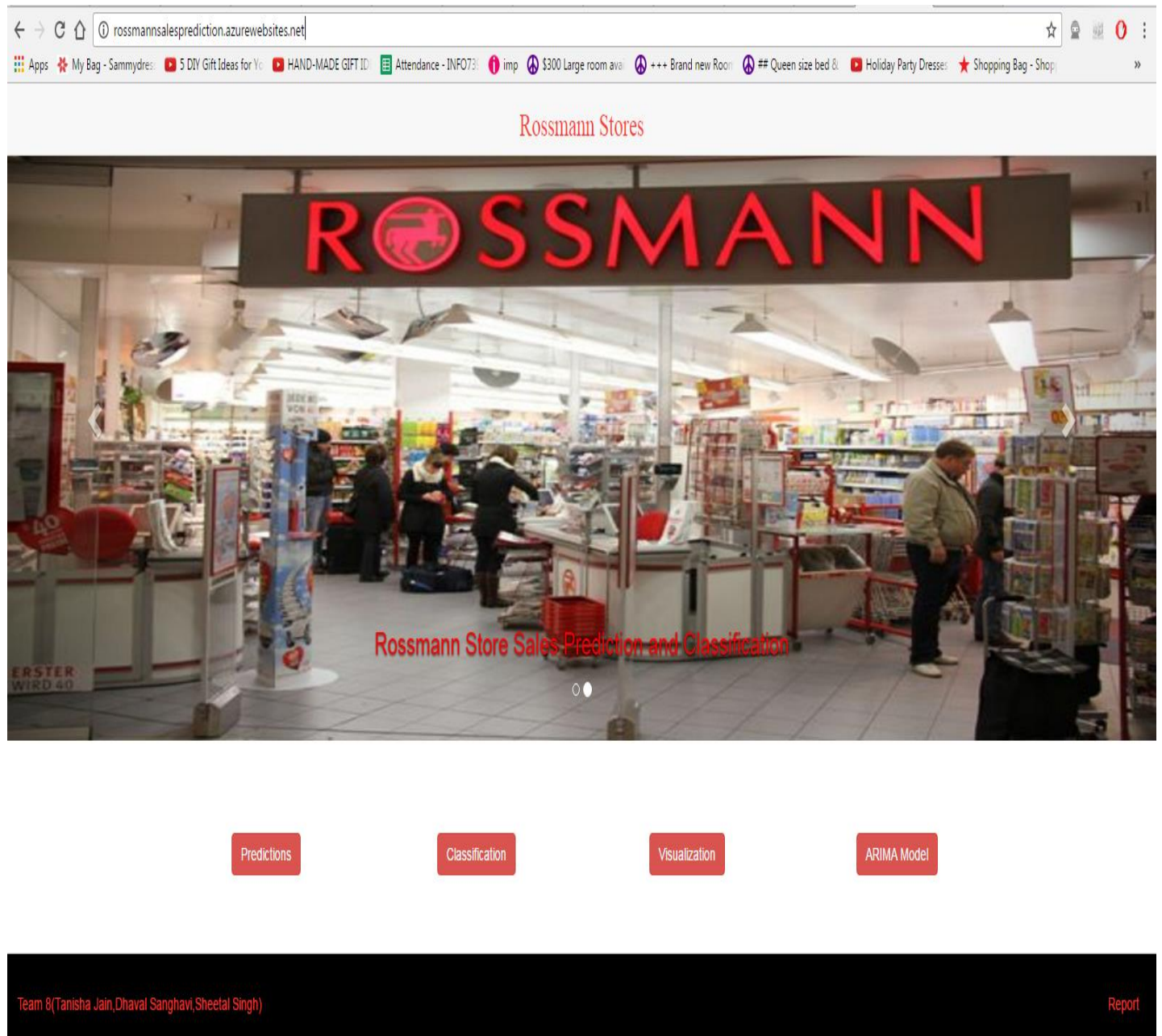


The following is the dataset obtained by training the clustering model with Rossmann sales dataset with  $K=2$ . The 'Assignments' column designates which cluster does the value of the particular row belongs to.



## 1.11 Web Application

### Main Page



## Sales Prediction

← → ↻ 🏠 ⓘ rossmannsalesprediction.azurewebsites.net/index.aspx

📱 Apps 🛒 My Bag - Sammydres: 📺 5 DIY Gift Ideas for Yo 📺 HAND-MADE GIFT ID 📅 Attendance - INFO73 ⓘ imp 🕒 \$300 Large room avail 🕒 +++ Brand new Room 🕒 ## Queen size bed & 📺 Holiday Party Dresses ⭐

### Sales Prediction

Store Type	b	Average Predicted Sales of a day in a month(in Euro):9345.06050159144
Year	2015	
Month	2	
DayofWeek	Weekday	
State Holiday	0	
School Holiday	0	
Assortment	a	
CompetitionDistance		
Promo2Indicator	0	
Promo	0	
<div>Predict the Sales</div>		
<div>Back</div>		

## Sales Classification

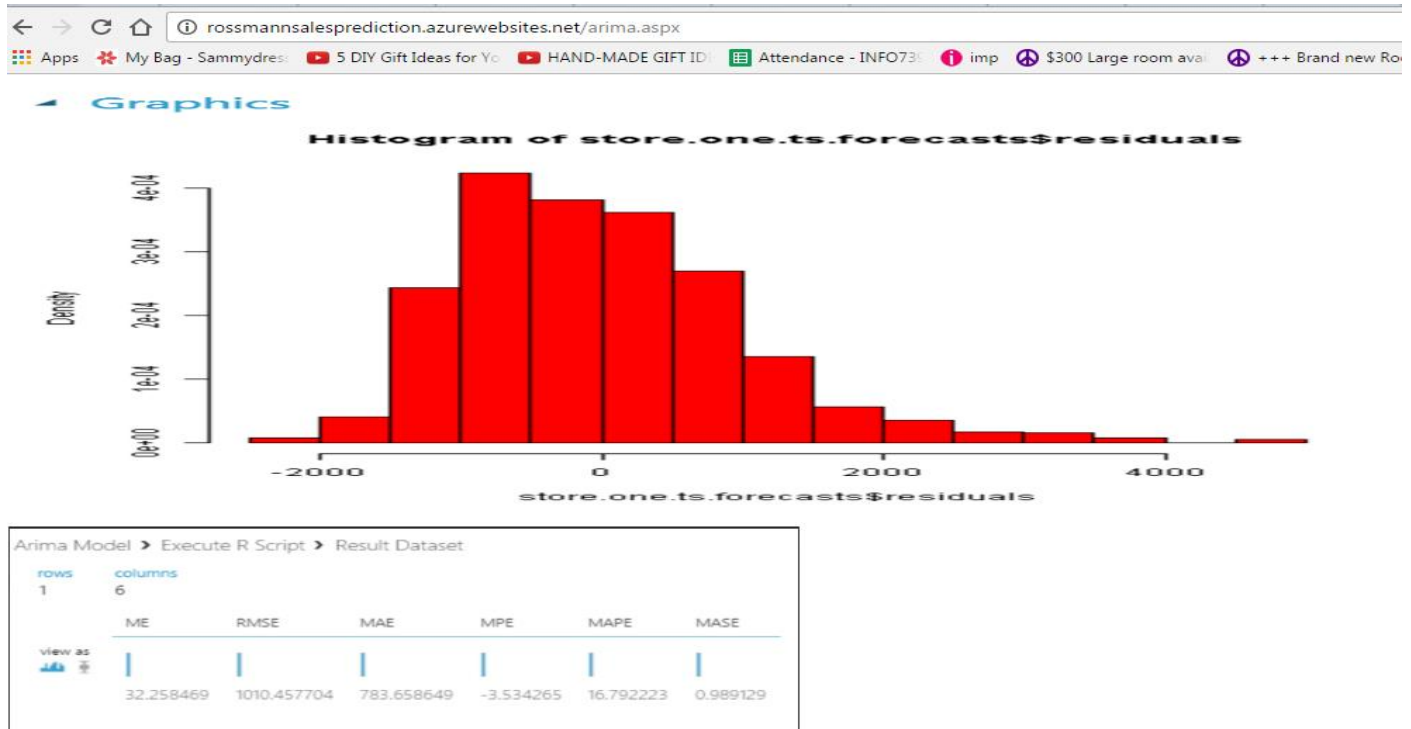
← → ↻ 🏠 ⓘ rossmannsalesprediction.azurewebsites.net/classification.aspx

📱 Apps 🛒 My Bag - Sammydres: 📺 5 DIY Gift Ideas for Yo 📺 HAND-MADE GIFT ID 📅 Attendance - INFO73 ⓘ imp 🕒 \$300 Large room avail 🕒 +++ Brand new Room 🕒 ## Queen size bed & 📺 Holiday Party Dresses ⭐

### Sales Classification

Store Type	d	Average Sales of a day in a month Classification:Below Average
Year	2016	
Month	2	
DayofWeek	Mon	
State Holiday	0	
School Holiday	0	
Assortment	b	
CompetitionDistance		
Promo2Indicator	0	
Promo	0	
<div>Classify the Sales</div>		
<div>Back</div>		

## ARIMA Model



### 1.13 References

1. <https://www.kaggle.com/c/rossmann-store-sales>
2. [https://en.wikipedia.org/wiki/Autoregressive\\_integrated\\_moving\\_average](https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average)
3. <https://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf>
4. [http://sutlib2.sut.ac.th/sut\\_contents/H99006.pdf](http://sutlib2.sut.ac.th/sut_contents/H99006.pdf)
5. <https://www.coursera.org/learn/machine-learning>