**Data Mining**

**Final Project Report**

Topic:

**Sales Prediction**

Done by:

**Rishya and Sanjay**

**Problem definition:**

Predicting the sales for October 2015 for every item and shop using various data mining modelling methods.

**Dataset:**

> str(merged\_all)

'data.frame': 2935849 obs. of 10 variables:

1 $ item\_id : int 0 1 1 1 1 1 1 2 2 3 ...

2 $ item\_category\_id : int 40 76 76 76 76 76 76 40 40 40 ...

3 $ item\_name : Factor w/ 22170 levels "! Ð’Ðž Ð’Ð›Ð\220Ð¡Ð¢Ð\230 Ð\24 $ item\_category\_name: Factor w/ 84 levels "Ð¡Ð»ÑƒÐ¶ÐµÐ±Ð½Ñ‹Ðµ",..: 47 80 80 5 $ shop\_id : int 54 55 55 55 55 55 55 54 54 54 ...

6 $ shop\_name : Factor w/ 60 levels "!Ð¯ÐºÑƒÑ‚Ñ\201Ðº Ð¢Ð¦ \"Ð¦ÐµÐ½Ñ‚ 7 $ date : Factor w/ 1034 levels "01.01.2013","01.01.2014",..: 28 $ date\_block\_num : int 20 21 15 20 18 15 19 19 22 18 ...

9 $ item\_price : num 58 4490 4490 4490 4490 4490 4490 58 58 100 ...

10 $ item\_cnt\_day : num 1 1 1 1 1 1 1 1 1 1 ...

The dataset consists of around 2.9 million data points of 10 variables as mentioned above.

**Data preparation and exploratory analysis**

1. **Cleaning the data:**
2. Handling missing values/ noisy data:

After a thorough analysis of the dataset, we found that there are no missing values in the dataset. However, there were few negative values for Item price and Item count.

item\_price item\_cnt\_day

Min. : -1.0 Min. : -22.000

1st Qu.: 249.0 1st Qu.: 1.000

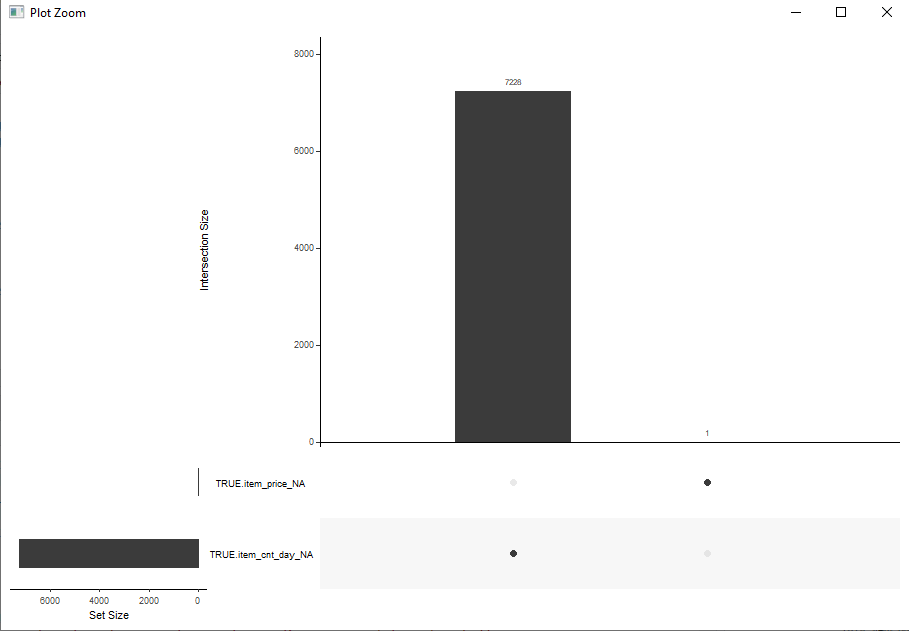
Median : 399.0 Median : 1.000

Mean : 890.9 Mean : 1.243

3rd Qu.: 999.0 3rd Qu.: 1.000

Max. :307980.0 Max. :2169.000

1. Graphical representation of the noisy data:



TRUE.item\_price TRUE.item\_cnt\_day

Min. : 0.07 Min. : 1.000

1st Qu.: 249.00 1st Qu.: 1.000

Median : 399.00 Median : 1.000

Mean : 884.66 Mean : 1.247

3rd Qu.: 999.00 3rd Qu.: 1.000

Max. :307980.00 Max. :1000.000

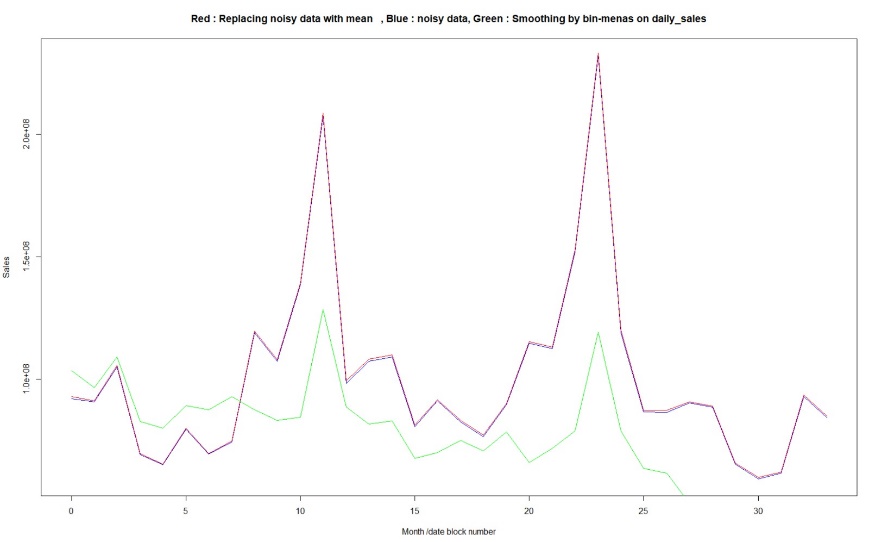
NA's :1 NA's :7228

The NA’s represent the count of the noisy data in the dataset.

1. Handling the noisy data:

The missing values were populated with the mean of the Item price and the mean of the Item count and got rid of the NA’s.

1. Impact of the above step:



Replacing the NA’s with the mean didn’t alter the original data. This can be seen in the above graph.

The Blue graph represents the original dataset with noisy data.

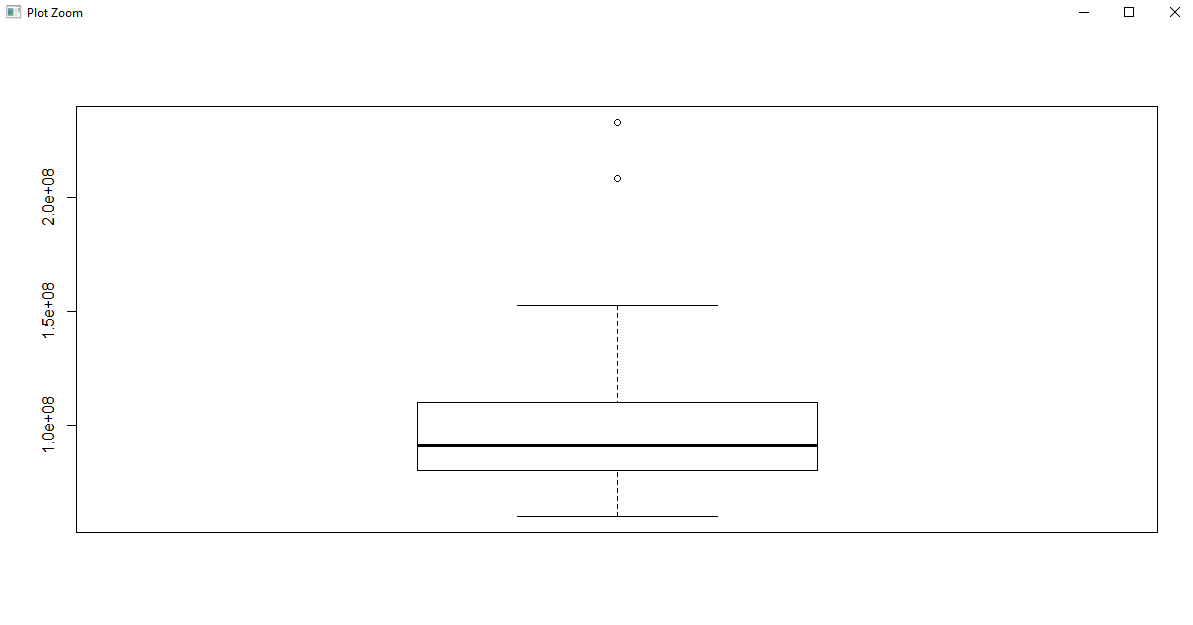
The Red graph represents the dataset after replacement with the mean value.

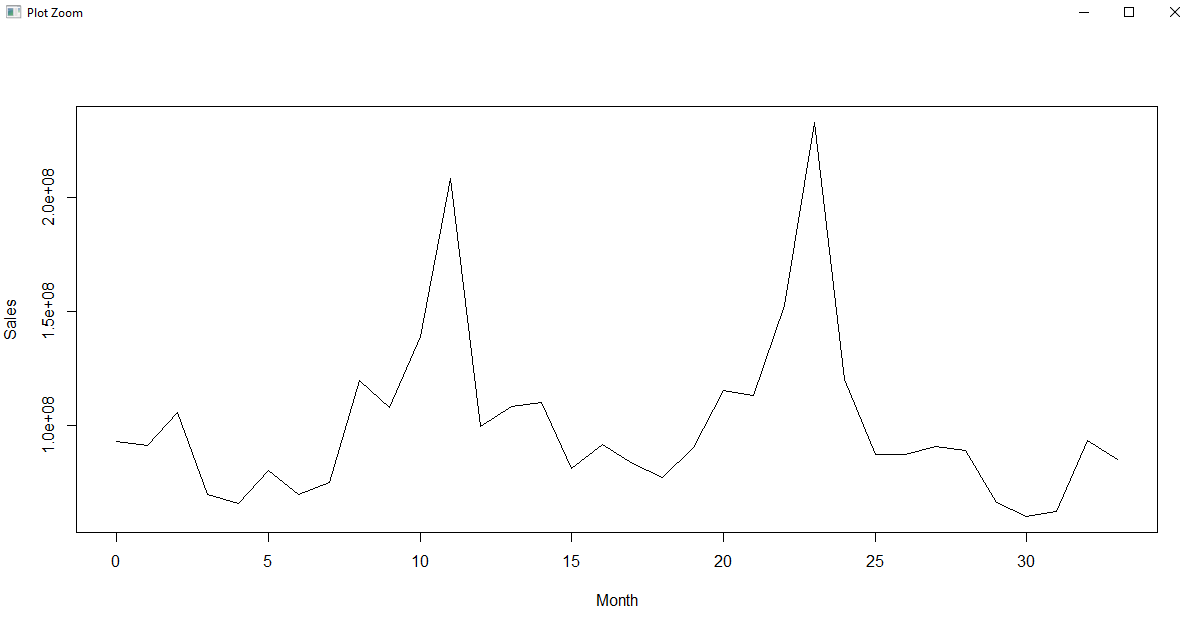
1. Outliers Detection:

Using the boxplot, the outliers are represented as follows:

1. Months of sales considered outliers:

We see that there are two months with exceptional sales, considered outliers.

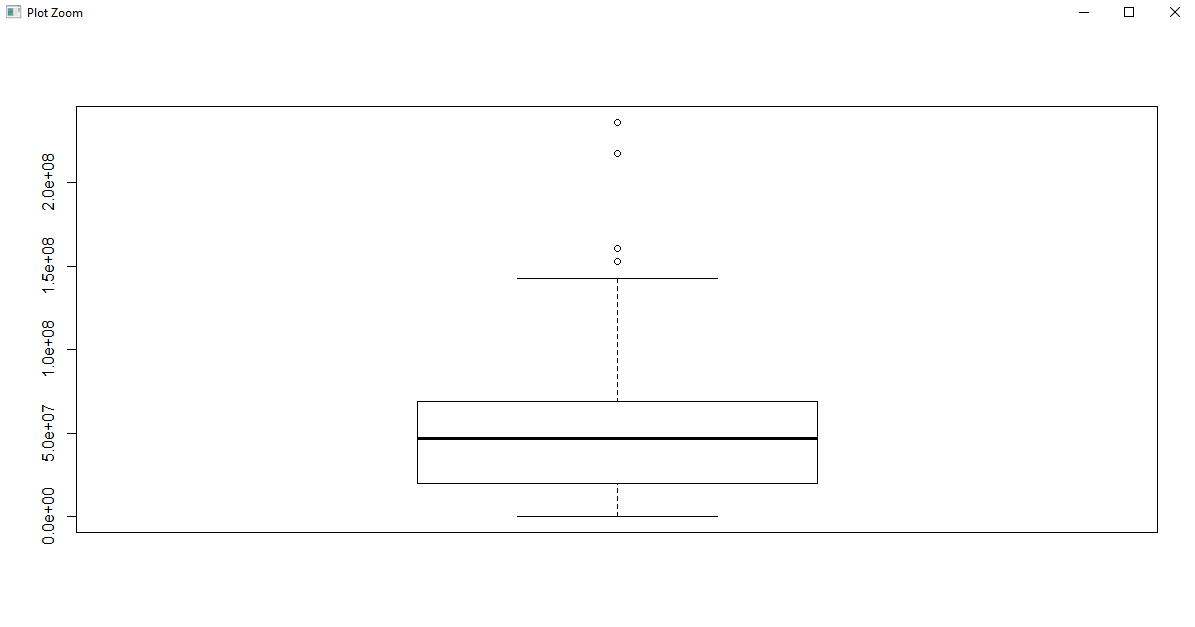


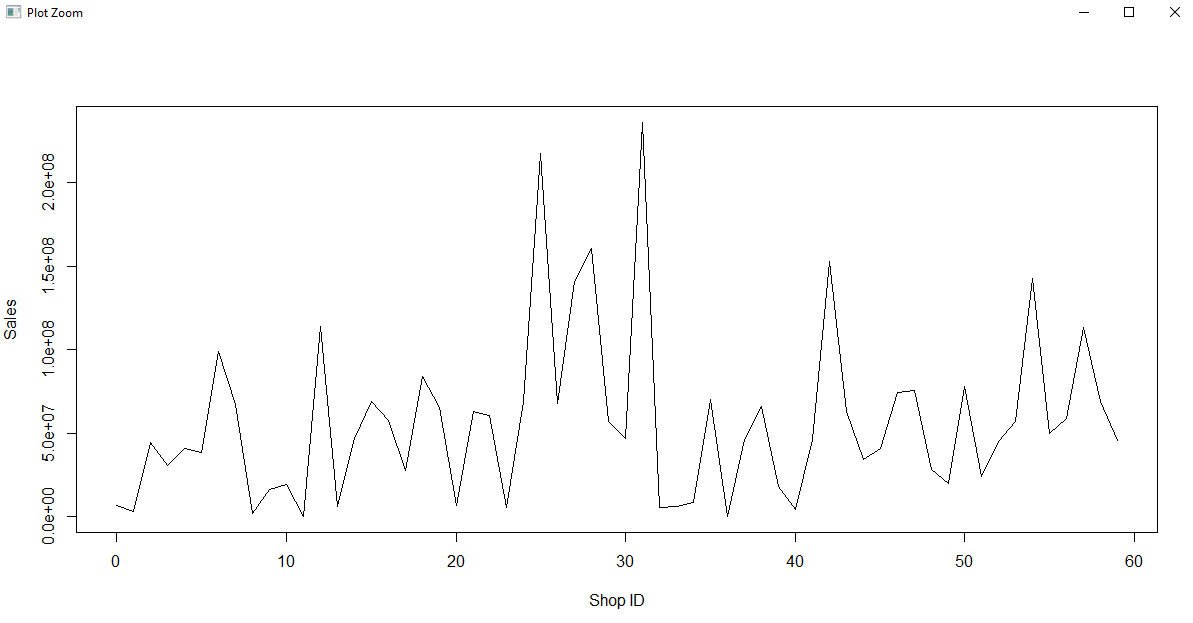


Two months: October and December are the months of sales which are outliers.

1. ShopIDs of sales considered outliers:

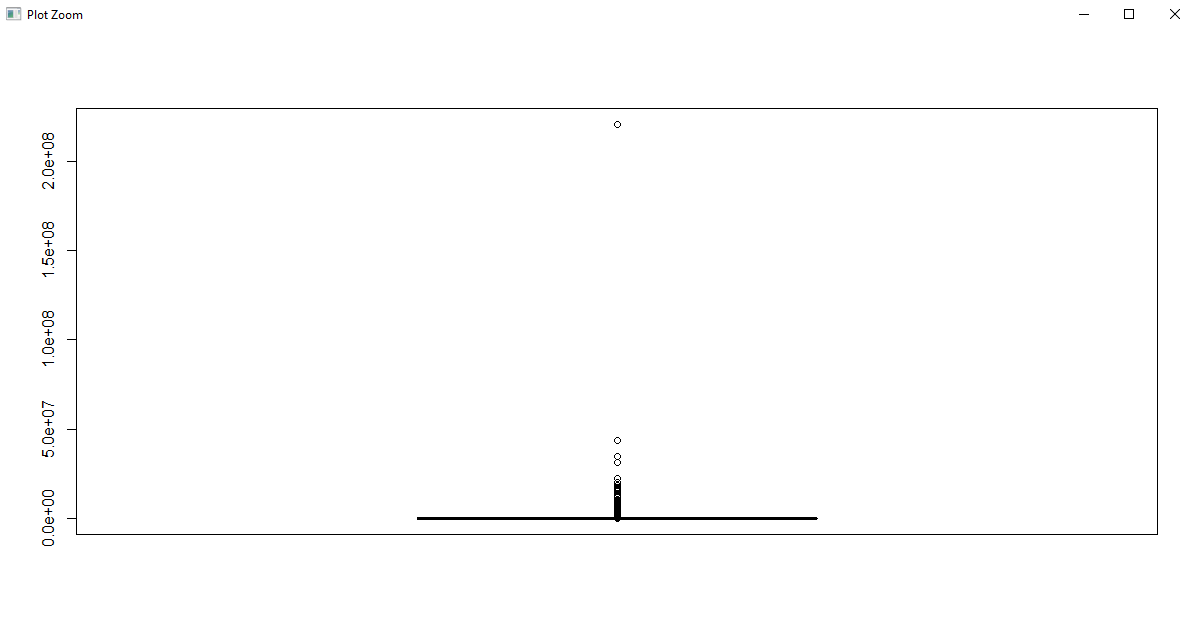
Four ShopIDs have exceptional sales, considered outliers.

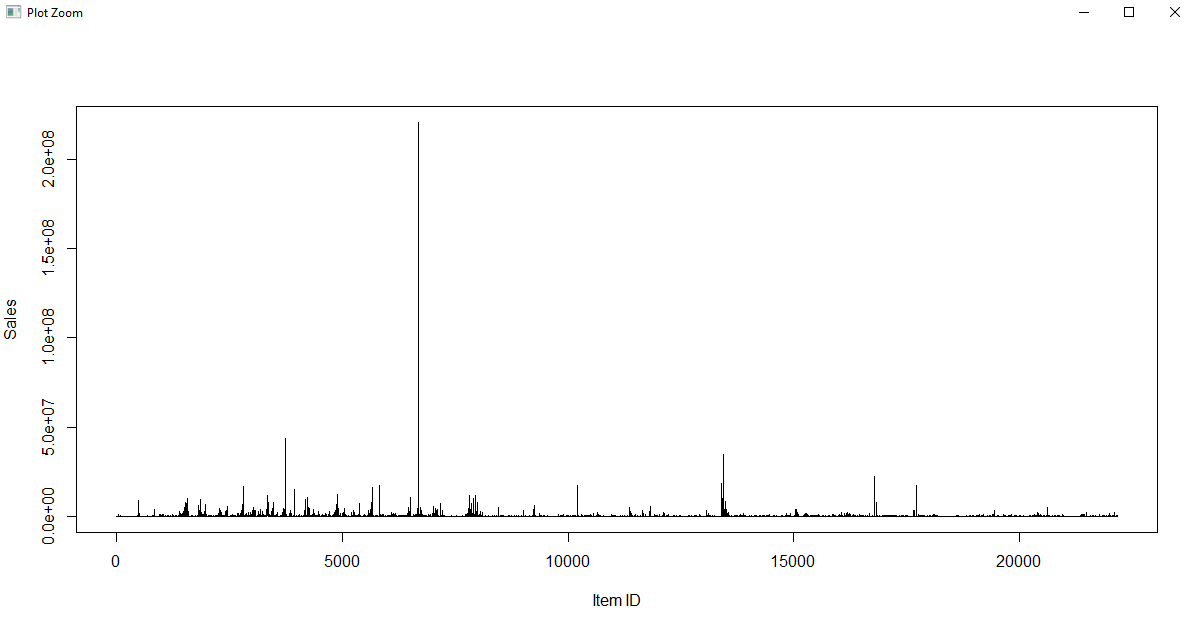




1. ItemIDs of sales as outliers:

There are quite a few outliers present in the sales w.r.t ItemIDs.





1. Feature selection/Engineering:

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive.

In this project we worked on feature engineering by creating new features (new columns):

1. Daily sales value : Count \* Item Price
2. Smoothing the daily sales value by bin-means

N = 200,000



1. Removed the columns that didn’t have an impact on modeling:

* Item name
* Shop name
* Item category name

There are columns like ItemID, ShopID which are used to instead of the above mentioned columns.

1. Modeling:

Data modeling refers to a group of processes in which multiple sets of data are combined and analyzed to uncover relationships or patterns. The goal of data modeling is to use past data to inform future efforts.

Various modeling techniques can be employed to predict the sales for the month of October 2015 as follows:

1. Linear Regression
2. K-Nearest Neighbor
3. Random Forest
4. Naïve Bayes
5. K-Means
6. Decision tree methods

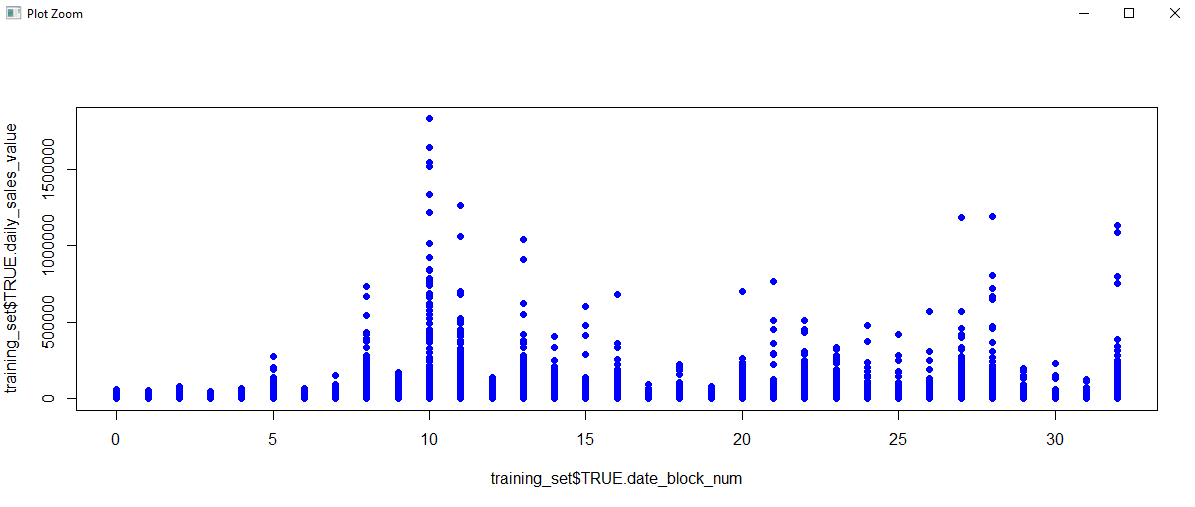
Of the above, we employed a. Linear Regression and b. K-Nearest Neighbor to predict the sales for October 2015.

1. Linear Regression:

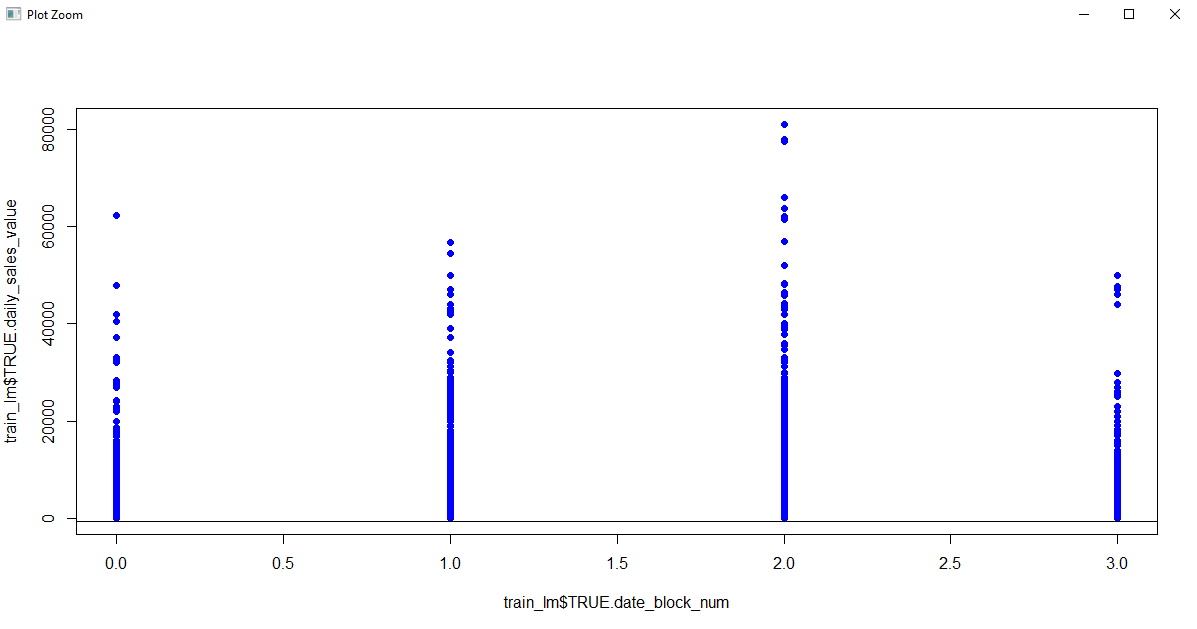
* + Step 1: Split the clean data Quarter-wise for efficient modelling purpose.
  + 
  + Step 2. Modelled the dataset for linear regression with respect to Daily Sales Value.
  + 
  + Step 3: Plotted the regression line.
  + Step 4: Predicted the output for the Test set i,e; October 2015.

Step 5: Plotted a comparison plot for actual values and predicted values.

The graph below is the scatter plot for the sales value for every month from January 2013 to September 2015.



The graph below displays the scatter plot for quarter 1, i.e. January 2013 to April 2013 with the regression line.

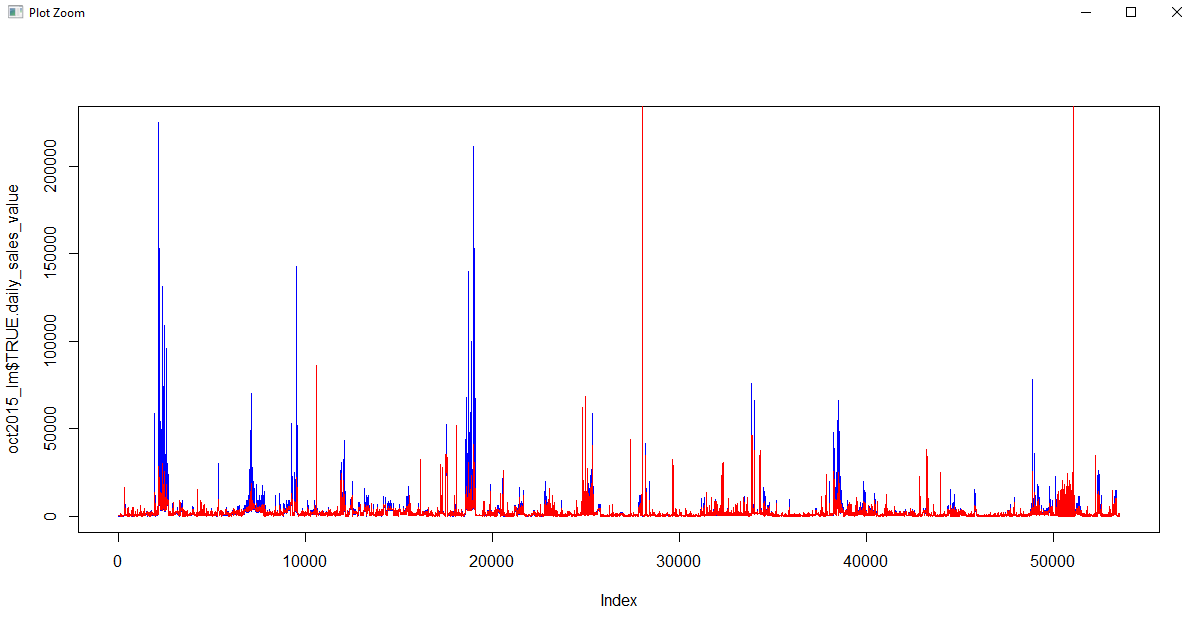


The graph here below represents the comparison of the actual data and predicted data by linear regression:

* 1. Plot a:

Plotting for linear regression predictions, where the actual values before engineered by smoothing bin-by-means method.

The Red lines in the graph represent the predicted values while the Blue lines represent the actual value.

Plot a

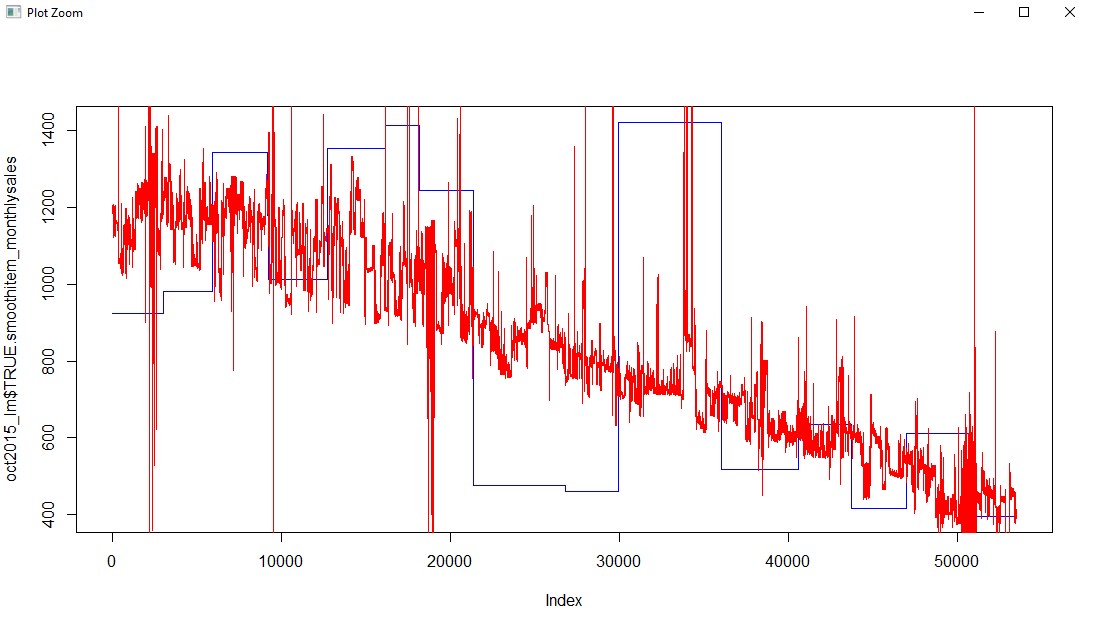
* 1. Plot b

The blue line represents the actual data of Oct 2015, while the red line represents the predicted data.

After a number of trials, the number of bins was selected was 200,000.

We observe that the actual and predicted values are in the same range and follow the same pattern.

The blue line in the graph is the representation of the actual sales value which are smoothed by binning by means method.

Plot b

1. K-Nearest Neighbors

Step 1: Normalize the entire dataset by creating a function.

normalize <-function(x) { return((x-min(x))/(max(x) - min(x))) }

norm\_data <- as.data.frame(lapply(knntraindata[,c(1,2,3,4,5,6,7,8)], normalize))

Step 2: Split the normalized data in train and test set on the date\_block \_num.

knn\_split <- split(norm\_data, norm\_data$TRUE.date\_block\_num < 1)

norm\_train\_knn <- data.frame(knn\_split["TRUE"])

norm\_test\_knn <- data.frame(knn\_split["FALSE"])

Step 3: Create a target dataset for daily\_sales.

knn\_split\_wo <- split(knntraindata, knntraindata$date\_block\_num <33)

wo\_norm\_train\_knn <- data.frame(knn\_split\_wo["TRUE"])

wo\_norm\_test\_knn <- data.frame(knn\_split\_wo["FALSE"])

train\_target <- wo\_norm\_train\_knn[,5]

Step 4: Determination of K

> k <- sqrt(2935849) > print(k) > [1] 1713.432

1. Validation:
2. Mean Square Error:

The mean squared error or mean squared deviation of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss.

* Linear Regression:
* With smoothing by bin-means:

> mse\_lm

[1] 753.3423

* Prior smoothing by bin-means:

> mse\_lm\_wo

[1] 28.09483

* K-Nearest Neighbors:

Root Mean Square Error:

The root-mean-square deviation or root-mean-square error is a frequently used measure of the differences between values predicted by a model or an estimator and the values observed.

1. Linear regression:

* With smoothing by bin-means:

> mse\_lm

[1] 753.3423

* Prior smoothing by bin-means:

> mse\_lm\_wo

[1] 28.09483

1. K-Nearest Means
2. Computation of Confidence Interval for both the models:
3. Linear Regression:

* For 80% Confidence Interval:

> CI\_80<-cbind(CIlower = mean(prediction\_lm) - 1.28 \* s / sqrtn,

+ CIupper = mean(prediction\_lm) + 1.28 \* s / sqrtn)

> CI\_80

CIlower CIupper

[1,] 845.3299 848.5863

* For 90% Confidence Interval:

> CI\_90<-cbind(CIlower = mean(prediction\_lm) - 1.64 \* s / sqrtn,

+ CIupper = mean(prediction\_lm) + 1.64 \* s / sqrtn)

> CI\_90

CIlower CIupper

[1,] 844.872 849.0443

* For 95% Confidence Interval:

> CI\_95<-cbind(CIlower = mean(prediction\_lm) - 1.96 \* s / sqrtn,

+ CIupper = mean(prediction\_lm) + 1.96 \* s / sqrtn)

> CI\_95

CIlower CIupper

[1,] 844.4649 849.4513

1. K-Nearest Neighbors
2. Comparison of the two models
3. Error:
4. Efficiency in training time (Scalability):