CASE STUDY REPORT

Group No: 19

Student Names: Ismail Ayub Saleh, Yiting Zhang

Executive Summary:

The goal of our study was to find the sales prediction for the different products at different shops for supermarket chain Bigmart. Also our goal was to learn new data mining techniques and enhance our knowledge of data mining while gaining real world insights on how data mining is done in industry. This Use Case is something which is very relevant to the data mining field in the age of ecommerce and retail.

Our data origin is from a competition on a website called Analyticsvidhya which caters to data mining enthusiasts and expert data miners alike.

We have two approaches to data processing in our use case. Our data had 18 % of rows with missing values. We dealt with it by replacing it with mean or zero in the first approach as we found appropriate. In the second approach we used data prediction and classification techniques along with the techniques of the first approach to replace the missing data. We ensure in both approaches that the rows are filled and there is no missing value. Also in the first approach we convert all categorical variables to numerical form. Also we used processing techniques like principle components analysis.

This is a regression problem. We used three data mining techniques Extreme gradient boosting (Objective=linear regression), Regression trees and variants of multiple linear regression.

We found out that performance wise Regression tree provided the best result. The results of other methods were very close. Only multiple linear regression, which included principle components with or without rotation as predictor variables performed worst as principle components accounted for very less variance. We recommend using regression tree for sales prediction tasks in the future. But for very large dataset we recommend using XGBOOST package as it has comparable performance and high speed of computation.

I. Background and Introduction

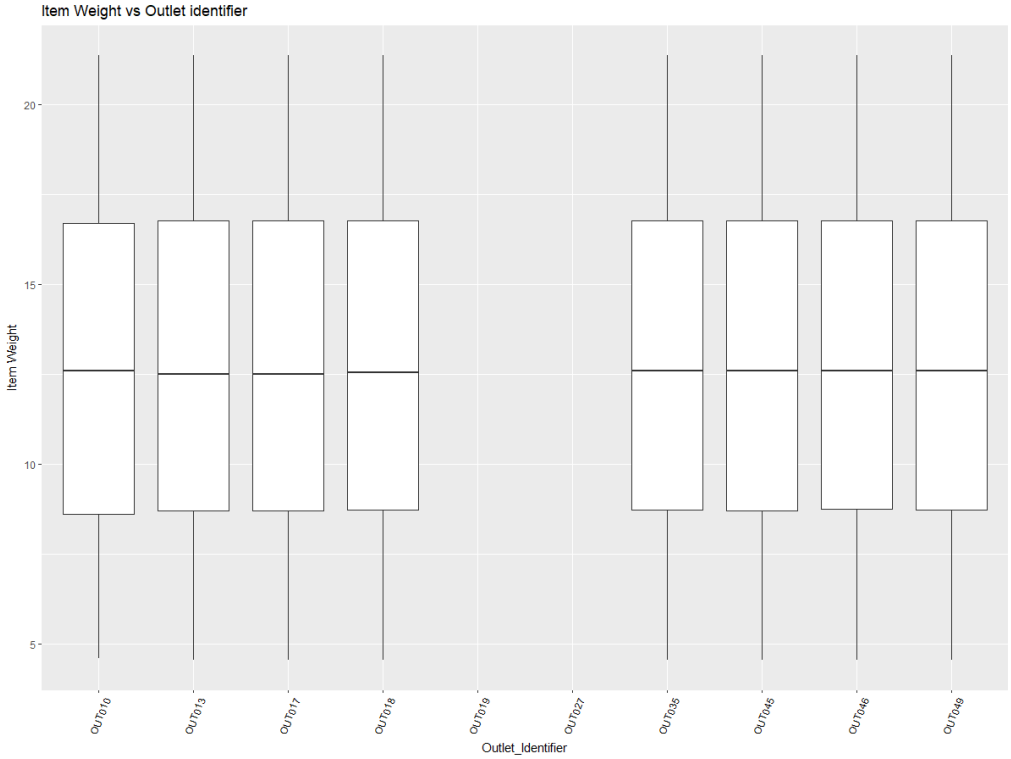
The data scientists at a Superstore chain have collected 2013 sales data for 1559 products across 10 stores in different cities. Those stores have different size, different type and different kinds of Items. Also, certain attributes of each product and store have been defined like the item weight, item fat content, item visibility, outlet size, location, type.

Item Identifier	Unique product ID
Item Weight	Weight of the product
Item Fat Content	Whether the product is low fat or not
Item Visibility	The percentage of total display area of all products in a store allocated to the particular product
Item Type	The category to which the product belongs
Item MRP	Maximum Retail Price of the product
Outlet Identifier	Unique store ID
Outlet Establishment Year	The year in which store was established
Outlet Size	The size of the store in terms of ground area covered
Outlet Location Type	Whether the outlet is just a grocery store or some sort of supermarket
Item Outlet Sales	Sales of the product in the particular store. This is the outcome variable to predicted

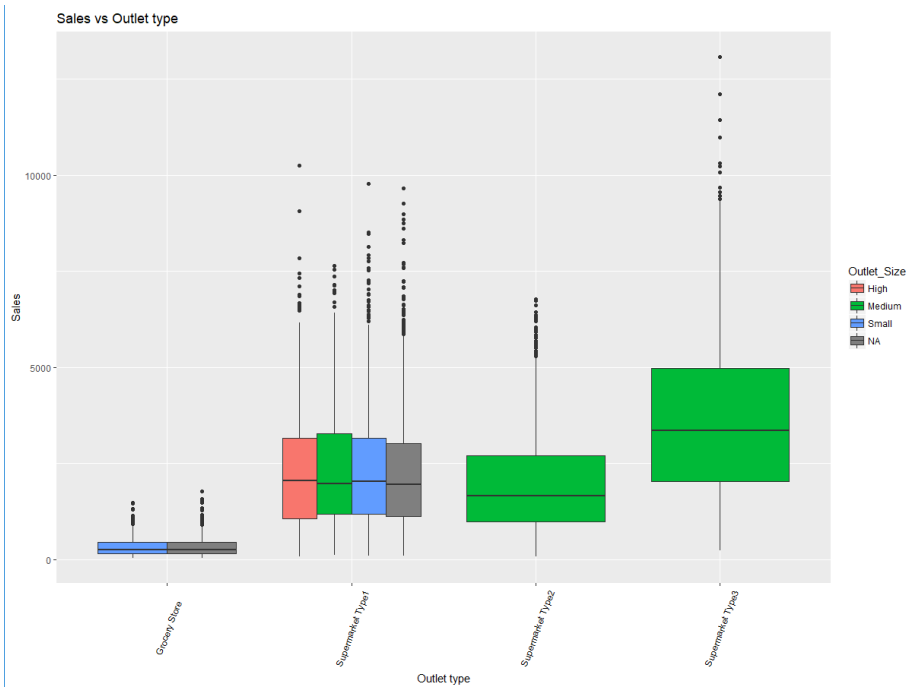
The aim is to build a predictive model and find out the sales of each product at a particular store. Using this model, the superstore chain will try to understand the properties of products and stores which play a key role in increasing sales. We study this because today in the age of retail and ecommerce sales predictions are one of the most important skills a data scientist can have.

II. Data Exploration and Visualization.

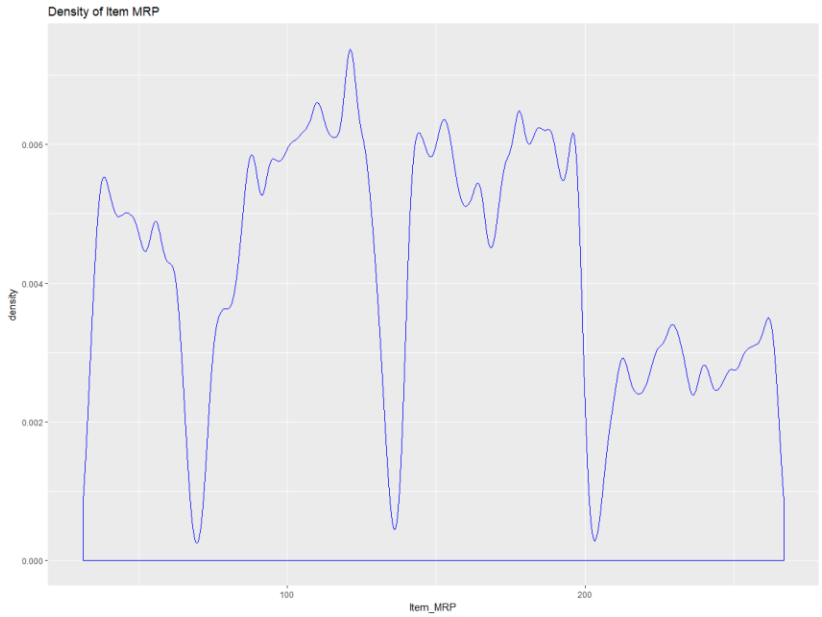
#Check the missing values.



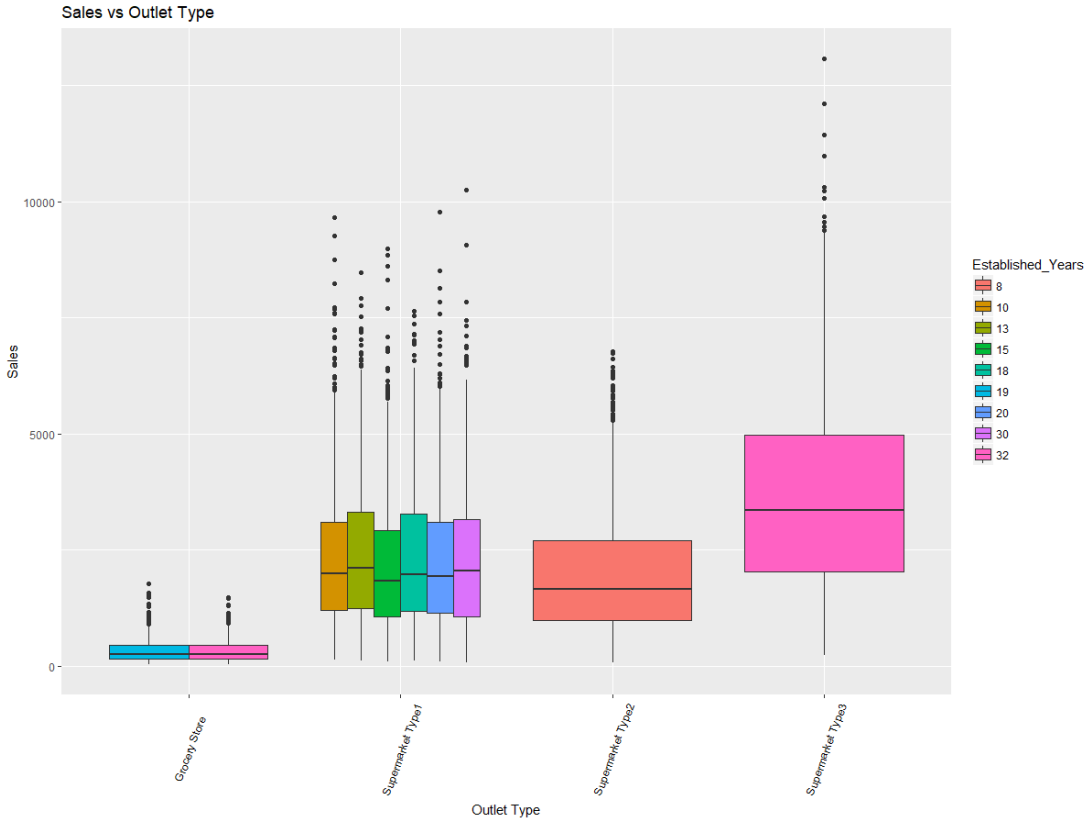
So the missing values are the Item Weight for outlet identifier of OUT019 and OUT027.The Item Weight has some missing values, so we can use the median to fill in the missing values of Item Weight. We have used zero and mean to replace missing values in First Approach. We used prediction techniques and classification techniques along with techniques of the first approach in the second approach of Data processing.



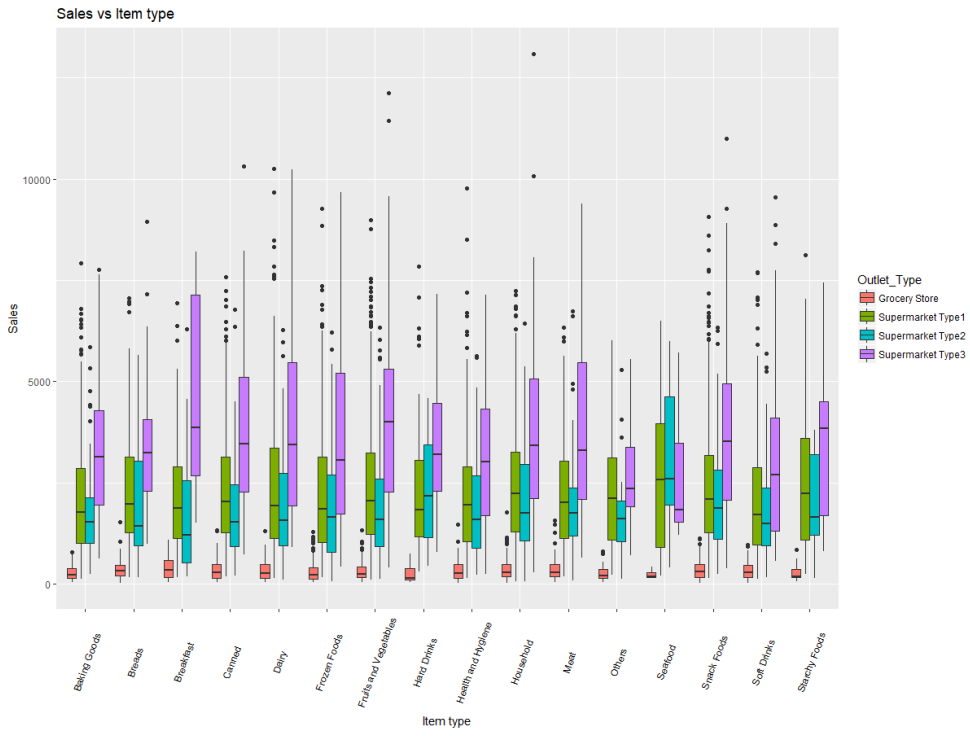
#There are missing values in Outlet_Size
#Of course a grocery store should be small size, and also the supermarkets in type 1 are most often classified as Small or medium

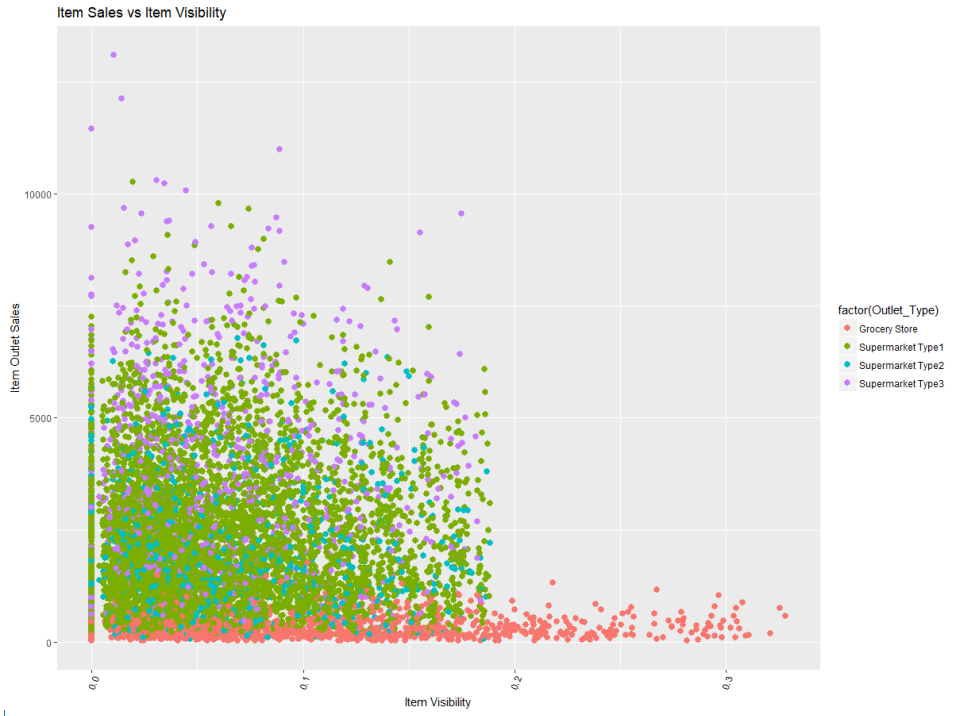


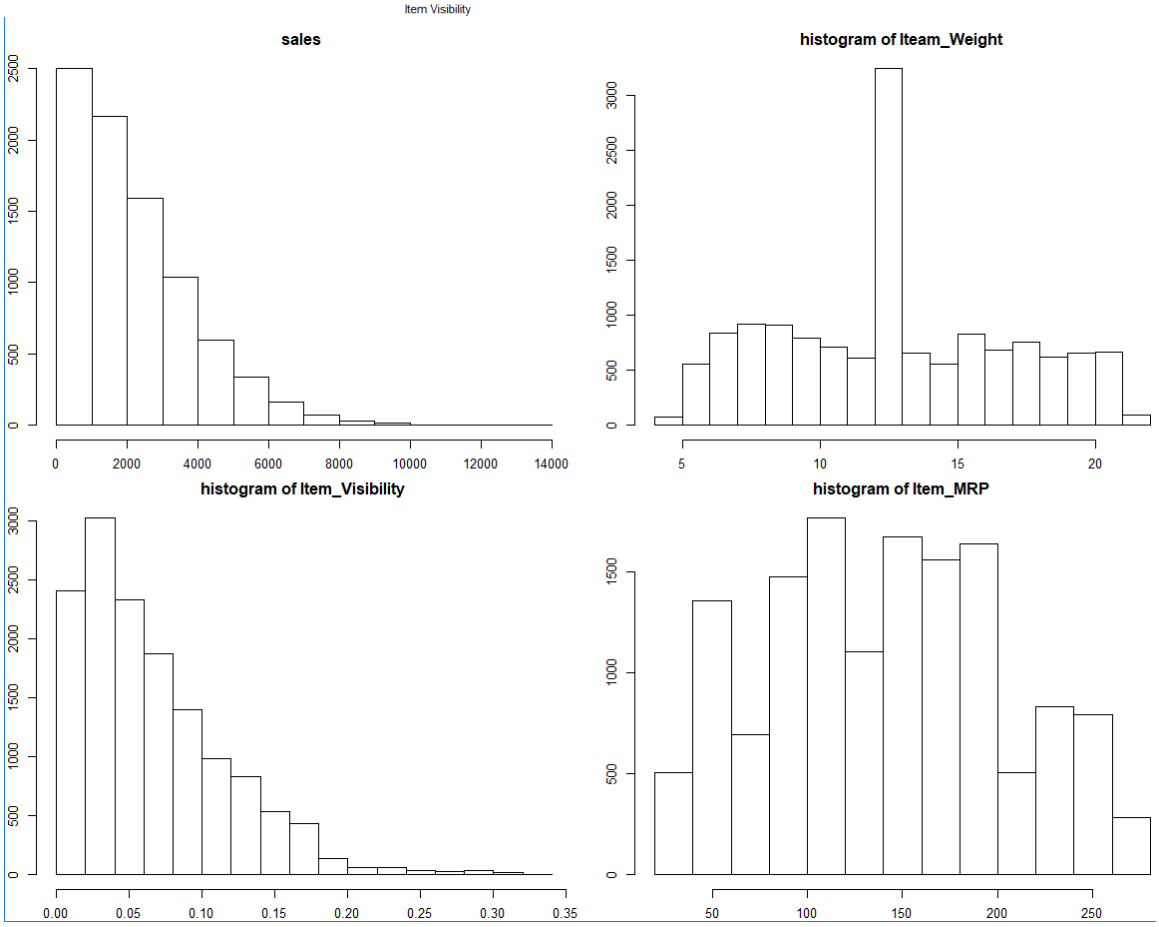
The MRP is located in four parts



Sales in the one type 2 supermarket appear a bit lower than supermarket type 1 and 3. Maybe it's because it is new established.




Item Sales vs Item Visibility
Item Outlet Sales
10000
5000
0
0.0
0.1
0.2
0.3
Item Visibility
factor(Outlet_Type)
Grocery Store
Supermarket Type1
Supermarket Type2
Supermarket Type3


sales
0
500
1000
1500
2000
2500
0
2000
4000
6000
8000
10000
12000
14000
histogram of Iteam_Weight
0
500
1000
1500
2000
2500
3000
5
10
15
20
histogram of Item_Visibility
0
500
1000
1500
2000
2500
3000
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
histogram of Item_MRP
0
500
1000
1500
50
100
150
200
250

III. Data Preparation and Preprocessing

For data preparation and preprocessing we have taken two approaches.

First Approach:-

The first approach is as follows. First of all split the data into train and test datasets and then tag the datasets by the train and test incorporating an extra column. Then combine the data again so we don't have to do the preparation twice. We replace the missing item weights by zero. We convert the Item fat content categorical variables into numeric, consolidating all the different labels for the same category into numerical form. We replace the missing item visibility with zero and take logarithmic function of the rest of the values multiplied by thousand, then replace any resultant infinite values with zero. We assign the Item type categorical variables with numerical values. We replace the missing Item MRP with mean item MRP. We assign different values to outlet establishment year the newer the higher value. Same way we encode outlet size with high having high value and small having less value. We encode Outlet location type and outlet type with numerical values. We replace missing item sales with zero. Then we convert Item outlet sales value to logarithm as that seems a more reasonable scale. Then we split the dataset into train and test again for our use and remove the unnecessary columns. We also tried a dummy variable approach rather than just giving numerical values but the numerical approach proved superior in results. This approach was taken due to the requirement of XGBOOST package we are using, which requires all numerical values. We will use this same approach for regression tree model.

Second Approach:-

In the second approach we check the missing values it is shown that 82 percentage is filled the rest is missing values. We convert the different label in Item Fat content into two categories. We use mean to replace the missing value in Item weight. We use multiple linear regression to replace the zeros in item visibility by predicting them. We use random forest to classify the missing outlet size. We confirm that all the values are filled and no value is missing. We carried out principal components analysis with and without rotation with both explaining about 52 % of variance. We also carried out all subsets regression to know which variables to select for the model.

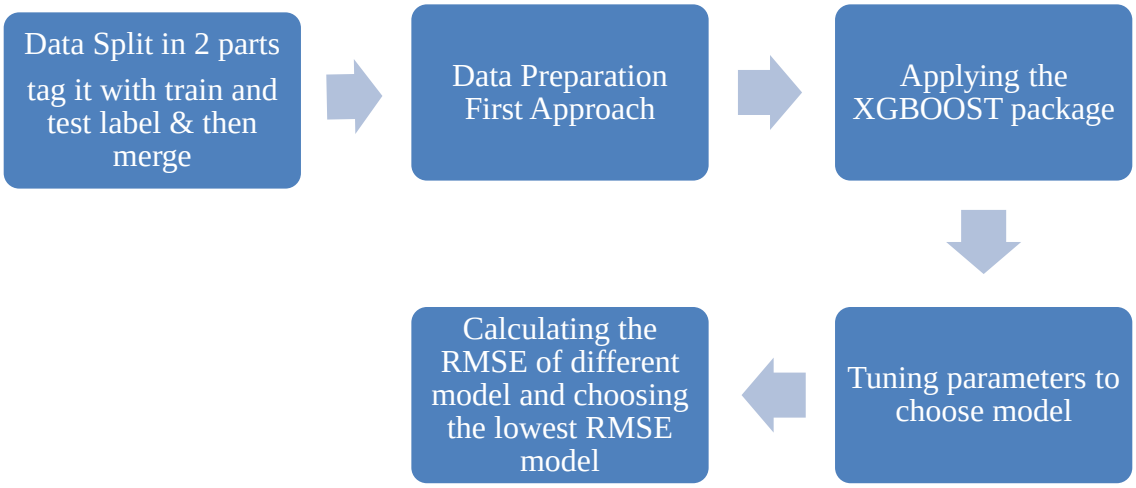
IV. Data Mining Techniques and Implementation

This problem is a regression problem. We have used extreme gradient boosting (XGBOOST), multiple linear regression and regression tree.

1). Extreme Gradient Boosting

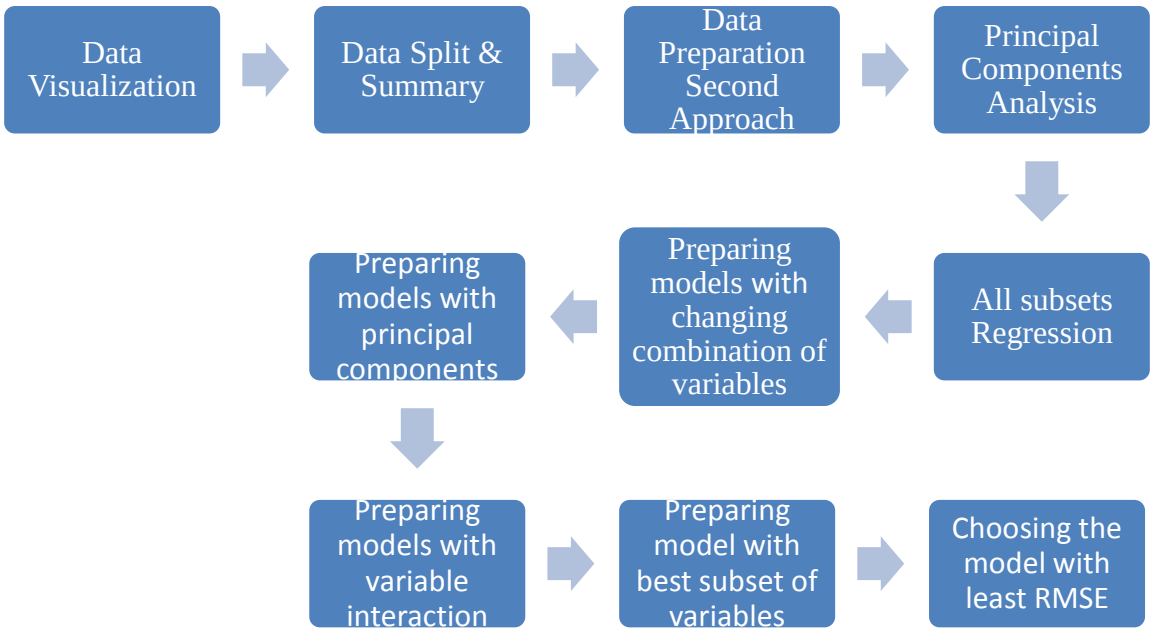
We used the first data preparation approach because XGBOOST works on numerical values only. We selected the objective as linear regression which results in changing default evaluation metric to root mean square error. It is an ensemble technique where new models are added to correct the errors of existing models.

It uses the gradient boosting decision tree algorithm. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models [1].



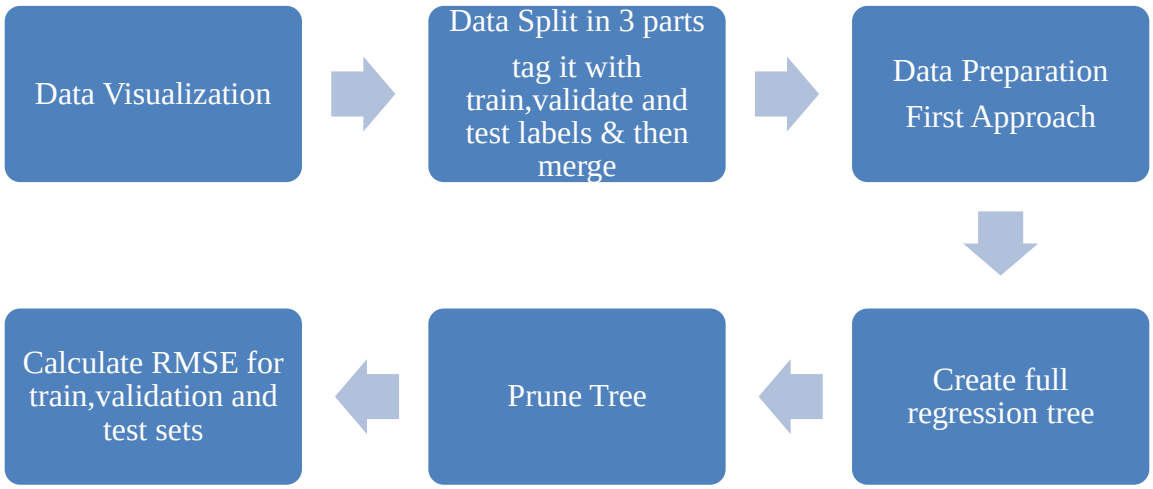
2) Multiple Linear Regression

As this is a regression problem this is a very straightforward method to make a model. Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data [2]. First we make a model by taking all the predictor variables, then a model by taking principal components created by Principal components Analysis with and without rotation. We also make a model by selecting the optimal subset of predictor variables by All subsets regression. We also try interaction between predictor variables in the model.



3) Regression Tree

All regression techniques contain a single output (response) variable and one or more input (predictor) variables. The output variable is numerical. The general regression tree building methodology allows input variables to be a mixture of continuous and categorical variables. A decision tree is generated when each decision node in the tree contains a test on some input variable's value. The terminal nodes of the tree contain the predicted output variable values. A Regression tree may be considered as a variant of decision trees, designed to approximate real-valued functions, instead of being used for classification methods [3]. .



V. Performance Evaluation

Performance Evaluation Table	
Method	RMSE
Multiple linear regression with all variables	1123.32
Multiple linear regression with PCA without rotation	1384.19
Multiple linear regression with PCA with rotation	1364
Multiple linear regression with interaction	1135
Multiple linear regression with best subset	1119
Optimum XG boost model RMSE	1123.38
Regression pruned tree RMSE(Train set)	1047
Regression pruned tree RMSE(validation set)	1111.04
Regression pruned tree RMSE(Test Set)	1097.16

Here we could clearly see that all methods are pretty close with the exception of multiple linear regression with PCA with and without rotation. Still we found that results were superior with regression tree. We select regression tree as a method of choice. Firstly we select it as it provides superior results. Secondly this method is easy to use, understand, produces rules that are easy to interpret & implement, further there is no need for transformation of variables, variable subset selection is automatic and this method can work without extensive handling of missing data.

VI. Discussion and Recommendation

We took two different approaches to data preparation and preprocessing with the first one based on a mixture of intuition and correlation analysis, which resulted in giving different numerical levels to categorical variables from correlation found from data visualization result and some numerical levels were given by common sense decision. In the second approach we summarized the data to study missing values and filled the missing values by using prediction techniques like multiple linear regression and classifying techniques like random forest. Advantages of regression tree are given in performance evaluation section its shortcomings are it may not perform well, where there is structure in the data that is not well captured by horizontal or vertical splits. Since the process deals with one variable at a time, there is no way to capture interactions between variables. In the other methods if we can tune the variables we may improve XGBOOST performance to be better than Regression tree. When there are large datasets XGBOOST'S speed and performance will help. The multiple linear regression is advantageous in that it helps us in evaluating the relative influence of different predictor variables. It is susceptible to mistakes due to incomplete or less data or falsely concluding that a correlation is the causation [4]. To improve we can try to get a larger and more complete dataset may be by adding other variables. We can also try logistic regression and polynomial regression to see what results we get and if we can improve the model.

VII. Summary

So to summarize, our aim of predicting sales for different products at different shops was fulfilled and we got a satisfactory result. We split the dataset, evaluated the missing data, converted variables and used the two data preparation approaches to replace missing values. In the second approach we used data prediction and classification techniques like multiple linear regression and random forest to predict and classify missing data. Further in multiple linear regression models we used preprocessing techniques to reduce the number of predictor variables by principle components analysis, carried out all subsets regression and then applied different data mining techniques as suitable to the two different data preparation approach we took. We evaluated the models on parameter of RMSE and selected the best model(pruned regression tree) with RMSE value 1097.16

Appendix: R Code for use case study

Please show the R code you generated for the use case study. Please do not show results here, only the code.

Data Visualization code example(not the full code) :-

```
library(ggplot2)
ggplot(Train, aes(Item_Type, Item_Weight)) +
  geom_boxplot() +  theme(axis.text.x = element_text(angle = 70, vjust = 0.5, color = "black")) +
  xlab("Item Type") +   ylab("Item Weight") +   ggtitle("Item Weight vs Item Type")

ggplot(Bigmart[1:nrow(Train),], aes(x = Outlet_Type, y = Item_Outlet_Sales, fill = Outlet_Size)) +
  geom_boxplot() +  theme(axis.text.x = element_text(angle = 70, vjust = 0.5, color = "black")) +
  xlab("Outlet type") +   ylab("Sales") +   ggtitle("Sales vs Outlet type")

Bigmart$Established_Years <- as.factor(2017-Bigmart$Outlet_Establishment_Year)
Bigmart <- select(Bigmart, -c(Outlet_Establishment_Year))

ggplot(Bigmart[1:nrow(Train),], aes(x = Outlet_Type, y = Item_Outlet_Sales, fill = Established_Years)) +
  geom_boxplot() +  theme(axis.text.x = element_text(angle = 70, vjust = 0.5, color = "black")) +
  xlab("Outlet Type") +   ylab("Sales") +   ggtitle("Sales vs Outlet Type")

par(mfrow=c(2,2))
hist( Bigmart$Item_Outlet_Sales, main = "sales ")
hist(Bigmart$Item_Weight, main="histogram of Iteam_Weight")
hist(Bigmart$Item_Visibility, main="histogram of Item_Visibility")
hist(Bigmart$Item_MRP, main="histogram of Item_MRP")
```

XGBOOST MODEL

```
Traini <- read.csv("Train.csv")
train_sample1 <- sample(8523,5114)
Train <- Traini[train_sample1,]
Test <- Traini[-train_sample1,]

Train <- mutate(Train, Segment="Train")
Test <- mutate(Test, Segment="Test")
DT <- rbind(Train, Test)

# ====================== DATA PREPARATION ================

Mean_Item_Weight <- mean(DT$Item_Weight, na.rm=TRUE)

DT$Item_Weight <- replace(DT$Item_Weigh, is.na(DT$Item_Weight), 0)

# LF, Low Fat, low fat, Regular, reg to be corrected

DT$Item_Fat_Content <- as.numeric(recode(DT$Item_Fat_Content,
                                 "'reg'=0; 'Regular'=0; 'LF'=1; 'low fat'=1; 'Low Fat'=1;
                                 else=0",
                                 as.factor.result=FALSE))
# DT$Item_Visibility should probably be log()
DT$Item_Visibility <- replace(DT$Item_Visibility, is.na(DT$Item_Visibility), 0)
# !!!
```

```
DT$Item_Visibility <- log(DT$Item_Visibility * 1000)
DT$Item_Visibility <- replace(DT$Item_Visibility, is.infinite(DT$Item_Visibility), 0)

# recode DT$Item_type to numeric
# change this to numeric later!
DT$Item_Type <- as.numeric(recode(DT$Item_Type,
                          "'Baking Goods'=1; 'Breads'=2; 'Breakfast'=3; 'Canned'=4;
                          'Dairy'=5; 'Frozen Foods'=6; 'Fruits and Vegetables'=7; 'Hard Drinks'=8;  'Health and
Hygiene'=9; 'Household'=10; 'Meat'=11; 'Others'=12;
  'Seafood'=13; 'Snack Foods'=14; 'Soft Drinks'=15; 'Starchy Foods'=16;
   else=0",      as.factor.result=FALSE))
# recode DT$Item_MRP missing values with mean
Mean_Item_MRP <- mean(DT$Item_MRP, na.rm=TRUE)

DT$Item_MRP <- replace(DT$Item_MRP, is.na(DT$Item_MRP), Mean_Item_MRP)

# recode DT$Establishment_Year to numeric
# change this to numeric later!
DT$Outlet_Establishment_Year <- as.factor(DT$Outlet_Establishment_Year)

DT$Outlet_Establishment_Year <- as.numeric(recode(DT$Outlet_Establishment_Year,
                                    "'1985'=1;          '1987'=1;          '1997'=2;          '1998'=2;
'1999'=2; '2002'=3; '2004'=3;  else=4",
                                    as.factor.result=FALSE))
# recode DT$Outlet_Size
# change this to numeric later!
DT$Outlet_Size <- as.numeric(recode(DT$Outlet_Size,
                          "'Small'=0; 'Medium'=1; 'High'=2; else=0",
                          as.factor.result=FALSE))
# recode DT$Outlet_Location_Type
DT$Outlet_Location_Type <- as.numeric(recode(DT$Outlet_Location_Type,
                                "'Tier 1'=0; 'Tier 2'=1; 'Tier 3'=2; else=0",
                                as.factor.result=FALSE))

# recode DT$Outlet_Type
DT$Outlet_Type <- as.numeric(recode(DT$Outlet_Type,
                          "'Grocery Store'=0; 'Supermarket Type1'=1; 'Supermarket Type2'=2; ; 'Supermarket
Type3'=3 ; else=0",
                          as.factor.result=FALSE))
DT$Item_Outlet_Sales <- as.numeric(DT$Item_Outlet_Sales)
Mean_Item_Outlet_Sales <- mean(as.numeric(DT$Item_Outlet_Sales), na.rm=TRUE)
DT$Item_Outlet_Sales <- replace(DT$Item_Outlet_Sales, is.na(DT$Item_Outlet_Sales), 0)
Test_outlet_sales <- Test$Item_Outlet_Sales
TOS <- as.data.frame(Test_outlet_sales)

# logarithm
DT$Item_Outlet_Sales <- log(DT$Item_Outlet_Sales)
DT$Item_Outlet_Sales <- replace(DT$Item_Outlet_Sale, is.infinite(DT$Item_Outlet_Sale), 0)

# ----- Create final test and train data  ------------------------

Train <- DT[DT$Segment=='Train',]
Test <- DT[DT$Segment=='Test',]
Train_original <- DT[DT$Segment=='Train',]
Train_values <- Train$Item_Outlet_Sales
Test_ID_item <- Test$Item_Identifier
```

```
Test_ID_outlet <- Test$Outlet_Identifier
Test_ID_outlet_type <- Test$Outlet_Type

Columns <- c(colnames(Train)[12:13], 'Item_Identifier' , 'Outlet_Identifier') # exclude identifiers, segment and value

Train <- Train[!colnames(Train) %in% Columns]

Test <- Test[!colnames(Test) %in% Columns]
# ============================== END DATA PREPARATION ====================
# ---- xgboost ---------
XG_Train <- xgb.DMatrix(as.matrix(Train,label=Train_values))
XG_Test = xgb.DMatrix(as.matrix(Test))
param <- list(
  objective = 'reg:linear',
  eta = 0.1,
  gamma = 1,
  eval_metric = 'rmse' ,
  min_child_weight = 4,
  max_depth = 4,
  subsample = 0.85,
  colsample_bytree = 0.5,
  max_delta_step = 20
)
rounds <- 1500
XG_Model <- xgb.train(param, XG_Train,rounds)
XG_Prediction <- predict(XG_Model, XG_Test)
# convert back to original scale with exponential
# !!!
XG_Prediction <- exp(XG_Prediction)

XG_Prediction <- cbind(as.character(Test_ID_item), as.character(Test_ID_outlet), Test_ID_outlet_type , as.character(XG_Prediction))

colnames(XG_Prediction) <- c("Item_Identifier", "Outlet_Identifier", "Item_Outlet_type", "Item_Outlet_Sales")
xgf <- as.data.frame(XG_Prediction)
XG_Feature_names <- dimnames(Train)[[2]]
XG_importance_matrix <- xgb.importance(XG_Feature_names, model = XG_Model)

# print(XG_importance_matrix)
XG_Train_Prediction <- predict(XG_Model, XG_Train)
XGP <- as.data.frame(XG_Prediction)
XGP[,4] <- as.numeric(as.character( XGP[, 4] ))
XG_Error <- sqrt(sum(((XGP$Item_Outlet_Sales - TOS$Test_outlet_sales)^2))/length(TOS$Test_outlet_sales))
XG_Error_Total <- sum(((XGP$Item_Outlet_Sales- TOS$Test_outlet_sales)^2))

#test error RMSE= 1123.38
```

Regression Tree(Rest of code similar to above approach the difference in code and the model is given)

```
Traini <- read.csv("Train.csv")
spec = c(train = .5, test = .2, validate = .3)
g = sample(cut(
  seq(nrow(Traini)),
  nrow(Traini)*cumsum(c(0,spec)),
  labels = names(spec)
))

res = split(Traini, g)
sapply(res, nrow)/nrow(Traini)
addmargins(prop.table(table(g)))
set.seed(1)

Train <- res$train
Test <- res$test
Validate <- res$validate
Train <- mutate(Train, Segment="Train")
Test <- mutate(Test, Segment="Test")
Validate <- mutate(Validate, Segment="Validate")
DT <- rbind(Train, Test,Validate)

#split into 3 sets
#do not convert sales into log
# the rest of data preparation and separation same as above then,
library(rpart)
Tree <- rpart(Item_Outlet_Sales ~ . , data= Train,method = "anova",maxdepth=6 ,minbucket=1,cp=-1)
summary(Tree)
Tree_pred <-predict(Tree,Test)
Tree_pred<- as.data.frame(Tree_pred)
library(Metrics)
rmserror <- rmse(Test$Item_Outlet_Sales,Tree_pred)
treeOptimal <- prune(Tree,cp=Tree$cptable[which.min(Tree$cptable[,4]),1])
summary(treeOptimal)
Tree_pred01 <- predict(treeOptimal,Validate)
rmserror01 <-rmse(Validate$Item_Outlet_Sales,Tree_pred01)
Tree_pred1 <- predict(treeOptimal,Test)
Tree_pred1 <- as.data.frame(Tree_pred1)
rmserror1 <- rmse(Test$Item_Outlet_Sales,Tree_pred1)
#test error RMSE= 1118.32
#test error prune tree = 1106.73
#2nd run train error=1099.613 validate error=1125.66 test error=1098.83
```

#2nd Approach

```
train1 <- read.csv("Train.csv")
train_sample1 <- sample(8523,5114)
train <- train1[train_sample1,]
test <- train1[-train_sample1,]

# Check rows are all filled?
nR <- nrow(train)
nC <- sum(complete.cases(train))
```

```
nC/nR

# Check fat levels
train$Item_Fat_Content

# Convert different variations of fat to consistent values
train$Item_Fat_Content <- gsub("LF", "lowfat",train$Item_Fat_Content)
train$Item_Fat_Content <- gsub("low fat", "lowfat",train$Item_Fat_Content)
train$Item_Fat_Content <- gsub("Low Fat", "lowfat",train$Item_Fat_Content)
train$Item_Fat_Content <- gsub("reg", "Regular",train$Item_Fat_Content)
train$Item_Fat_Content <- as.factor(train$Item_Fat_Content)
summary(train$Item_Fat_Content)

# Using mean to replace the missing values in Item_Weight variable
MeanIW <- mean(train$Item_Weight,na.rm=TRUE)
train$Item_Weight[is.na(train$Item_Weight)] <- MeanIW

# Using regression to replace the zeros in Item_visibility variable
training <- train %>% filter(Item_Visibility != 0)
model <- lm(Item_Visibility ~ Item_Weight + Item_Fat_Content +
              Item_Type + Item_MRP +
              Outlet_Establishment_Year + Outlet_Size +
              Outlet_Location_Type + Item_Outlet_Sales,
            data = training)
train$Item_Visibility[train$Item_Visibility == 0] <-
  predict(model,newdata = train[train$Item_Visibility == 0,])

# Classify missing values in Outlet Size
set.seed(100)
train$Outlet_Size <- as.character(train$Outlet_Size)
OS <- subset(train, Outlet_Size != "")
spl <- sample.split(OS$Outlet_Size, SplitRatio = 0.8)
traino <- subset(OS, spl == TRUE)
testo <- subset(OS, spl == FALSE)
## Using Random Forest for classification
traino$Outlet_Size <- as.factor(traino$Outlet_Size)
testo$Outlet_Size <- as.factor(testo$Outlet_Size)
## Creating the model
Tree <- randomForest(Outlet_Size ~.-Item_Outlet_Sales -Item_Identifier,
                     data =  traino,nodesize = 25, ntree = 100)
## Predicting on the test set
Predtree <- predict(Tree, newdata = testo)
## Confusion matrix
table(testo$Outlet_Size, Predtree)
# Classify
train$Outlet_Size <- predict(Tree, newdata = train)

# Check rows are all filled? Yes they are
nR <- nrow(train)
nC <- sum(complete.cases(train))
nC/nR

# Get final summary
summary(train)
```

```
# Do same everything for test
test$Item_Fat_Content <- gsub("LF", "lowfat",test$Item_Fat_Content)
test$Item_Fat_Content <- gsub("low fat", "lowfat",test$Item_Fat_Content)
test$Item_Fat_Content <- gsub("Low Fat", "lowfat",test$Item_Fat_Content)
test$Item_Fat_Content <- gsub("reg", "Regular",test$Item_Fat_Content)
test$Item_Fat_Content <- as.factor(test$Item_Fat_Content)
MeanItem_Weight <- mean(test$Item_Weight[!is.na(test$Item_Weight)])
test$Item_Weight[is.na(test$Item_Weight)] <- MeanItem_Weight
testing1 <- test %>% filter(Item_Visibility != 0)
model1 <- lm(Item_Visibility ~ Item_Weight + Item_Fat_Content +
              Item_Type + Item_MRP +
              Outlet_Establishment_Year + Outlet_Size +
              Outlet_Location_Type,
            data = testing1)
test$Item_Visibility[test$Item_Visibility == 0] <-
  predict(model1,newdata = test[test$Item_Visibility == 0,])
set.seed(100)
test$Outlet_Size <- as.character(test$Outlet_Size)
St1 <- subset(test, Outlet_Size != "")
spl <- sample.split(St1$Outlet_Size, SplitRatio = 0.8)
train_outlet <- subset(St1, spl == TRUE)
test_outlet <- subset(St1, spl == FALSE)
train_outlet$Outlet_Size <- as.factor(train_outlet$Outlet_Size)
test_outlet$Outlet_Size <- as.factor(test_outlet$Outlet_Size)
SF <- randomForest(Outlet_Size ~.-Item_Identifier,
                   data =  test_outlet,nodesize = 25, ntree = 100)
PF <- predict(SF, newdata = test_outlet)
test$Outlet_Size <- predict(SF, newdata = test)

library(psych)
fa.parallel(train[,c(2,4,6,8)],fa="pc", n.iter = 100, show.legend = TRUE, main="scree plot with parallel
analysis")
abline(h=1,lwd=1,col="green")
pc <- principal(train[,c(2,4,6,8)],nfactors = 2,rotate="none")
pc
rc <- principal(train[,c(2,4,6,8)],nfactor=2,rotate="varimax")
rc
pc$scores
rc$scores
# Build the model
test_pred <- test
sales_model <- lm(Item_Outlet_Sales ~ rc$scores +  Item_Fat_Content + Outlet_Size +
                    Outlet_Location_Type + Outlet_Type,
                  data = train)
summary(sales_model)
fa.parallel(test[,c(2,4,6,8)],fa="pc", n.iter = 100, show.legend = TRUE, main="scree plot with parallel
analysis")
abline(h=1,lwd=1,col="green")
rc <- principal(test[,c(2,4,6,8)],nfactor=2,rotate="varimax")
rc
sales <- predict(sales_model, newdata = test_pred)
test_pred$Item_Outlet_Sales <- as.vector(sales)

# Rename the predicted sales column
names(test_pred)[12] <- "Item_Outlet_Sales"
```

```
answer = test_pred[, c("Item_Identifier", "Outlet_Identifier", "Item_Outlet_Sales")]
train_Pred1 <- predict(sales_model,train)
error <- sqrt(sum(((test_pred$Item_Outlet_Sales - test$Item_Outlet_Sales)^2))/length(test_pred$Item_Outlet_Sales))
Error_Total <- sum(((sales- test$Item_Outlet_Sales)^2))
#Rms error = 1384.19 with pca wo rotation
#1364 with rotation
sales_model <- lm(Item_Outlet_Sales ~ Item_Weight + Item_Fat_Content +
                  Item_Visibility + Item_MRP +
                  Outlet_Establishment_Year + Outlet_Size +
                  Outlet_Location_Type + Outlet_Type,
                data = train) # rest of code same as above

#Without principal component analysis #Rms error = 1123.32

sales_model <- lm(Item_Outlet_Sales ~   Item_MRP + Item_Type +
                  Outlet_Type,
                data = train)     # model with variables as shown by subsets  selection shown below
library(leaps)
leaps <- regsubsets(Item_Outlet_Sales ~ Item_Weight + Item_Fat_Content +
                    Item_Visibility + Item_MRP  + Item_Type+
                    Outlet_Establishment_Year + Outlet_Size +
                    Outlet_Location_Type + Outlet_Type,,
                  data = train,nbest=2)
plot(leaps,scale="adjr2")
summary(sales_model)

sales_model <- lm(Item_Outlet_Sales ~  Item_Fat_Content  + Item_MRP  +
                  Outlet_Establishment_Year +
                  Outlet_Size:Outlet_Type,
                data = train)
#Rms error = 1123.32 with all variables
#1137 with interaction combination 1
#1135 with interaction combination 2
# with interaction there is a minor difference
```

References:-
1).https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
2). http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm
3). https://www.solver.com/regression-trees
4).https://sciencing.com/advantages-disadvantages-multiple-regression-model-12070171.html