Analysis of Big Mart Dataset

Shubham Agrawal

15 March 2019

Introduction

The data scientists at BigMart have collected sales data for 1559 products across 10 stores in different cities for the year 2013. Now each product has certain attributes that sets it apart from other products. Same is the case with each store.

The aim is to build a predictive model to find out the sales of each product at a particular store so that it would help the decision makers at BigMart to find out the properties of any product or store, which play a key role in increasing the overall sales.

You can read the problem statement and download the datasets from this link: Analytics Vidhya

Stage 1: Hypothesis Generation

This is an important step in the process of analyzing data. It involves understanding the problem and making some hypothesis about what could potentially have a good impact on the outcome. NOTE: This is done BEFORE looking at the data.

Some of the hypothesis are:

1. **City type:** Stores located in urban or Tier 1 cities should have higher sales because of the higher income levels of people there.
2. **Population Density:** Stores located in densely populated areas should have higher sales because of more demand. Store Capacity: Stores which are very big in size should have higher sales as they act like one-stop-shops and people would prefer getting everything from one place
3. **Competitors:** Stores having similar establishments nearby should have less sales because of more competition.
4. **Brand:** Branded products should have higher sales because of higher trust in the customer.
5. **Packaging:** Products with good packaging can attract customers and sell more.
6. **Utility:** Daily use products should have a higher tendency to sell as compared to the specific use products.
7. **Family Size:** More the number of family members, more amount will be spent by a customer to buy products.
8. **Annual Income:** Higher the annual income of a customer, customer is more likely to buy high cost products. Past Purchase History: Availablity of this information can help us to determine the frequency of a product being purchased by a user.
9. **Economic Growth:** If the current economy shows a consistent growth, per capita income will rise, therefore buying power of customers will increase.

(For more information, visit the following page.)

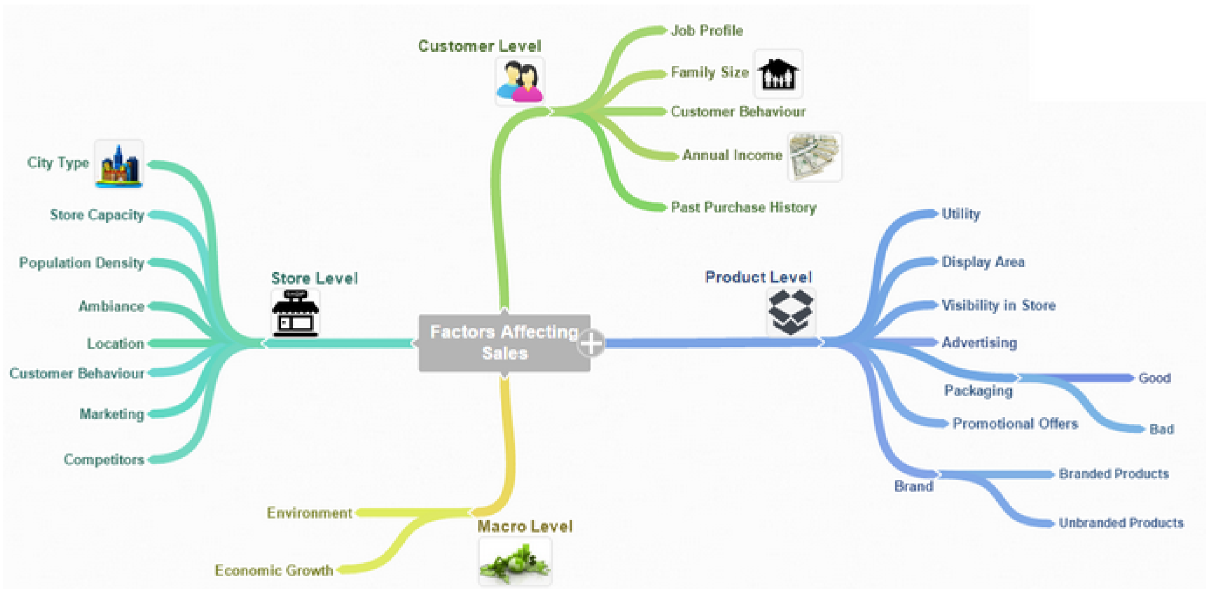


Figure 1: *(Source: Analytics Vidhya)*

Stage 2: Exploratory Data Analysis

In this section, we will explore the data through visualization and other methods and make some inference about our data. Further, we will look out for any irreregularities in the dataset, so that we can correct them in the pre-processing stage. First, load libraries and read data from the file.

```r
library(data.table)
library(dplyr)
library(ggplot2)
library(caret)
library(corrplot)
library(xgboost)
library(cowplot)
library(gridExtra)
library(dummies)
library(stringr)


train <- read.csv('Train.csv')
test <- read.csv('Test.csv')

str(train)
```

```
## 'data.frame':    8523 obs. of  12 variables:
##  $ Item_Identifier          : Factor w/ 1559 levels "DRA12","DRA24",..: 157 9 663 1122 1298 759 697
##  $ Item_Weight              : num  9.3 5.92 17.5 19.2 8.93 ...
##  $ Item_Fat_Content         : Factor w/ 5 levels "LF","low fat",..: 3 5 3 5 3 5 5 3 5 5 ...
##  $ Item_Visibility          : num  0.016 0.0193 0.0168 0 0 ...
##  $ Item_Type                : Factor w/ 16 levels "Baking Goods",..: 5 15 11 7 10 1 14 14 6 6 ...
##  $ Item_MRP                 : num  249.8 48.3 141.6 182.1 53.9 ...
##  $ Outlet_Identifier        : Factor w/ 10 levels "OUT010","OUT013",..: 10 4 10 1 2 4 2 6 8 3 ...
```

```
##  $ Outlet_Establishment_Year: int  1999 2009 1999 1998 1987 2009 1987 1985 2002 2007 ...
##  $ Outlet_Size              : Factor w/ 4 levels "","High","Medium",..: 3 3 3 1 2 3 2 3 1 1 ...
##  $ Outlet_Location_Type     : Factor w/ 3 levels "Tier 1","Tier 2",..: 1 3 1 3 3 3 3 3 2 2 ...
##  $ Outlet_Type              : Factor w/ 4 levels "Grocery Store",..: 2 3 2 1 2 3 2 4 2 2 ...
##  $ Item_Outlet_Sales        : num  3735 443 2097 732 995 ...
```

```r
str(test)
```

```
## 'data.frame':    5681 obs. of  11 variables:
##  $ Item_Identifier          : Factor w/ 1543 levels "DRA12","DRA24",..: 1104 1068 1407 810 1185 462 
##  $ Item_Weight              : num  20.75 8.3 14.6 7.32 NA ...
##  $ Item_Fat_Content         : Factor w/ 5 levels "LF","low fat",..: 3 4 3 3 5 5 5 3 5 3 ...
##  $ Item_Visibility          : num  0.00756 0.03843 0.09957 0.01539 0.1186 ...
##  $ Item_Type                : Factor w/ 16 levels "Baking Goods",..: 14 5 12 14 5 7 1 1 14 1 ...
##  $ Item_MRP                 : num  107.9 87.3 241.8 155 234.2 ...
##  $ Outlet_Identifier        : Factor w/ 10 levels "OUT010","OUT013",..: 10 3 1 3 6 9 4 6 8 3 ...
##  $ Outlet_Establishment_Year: int  1999 2007 1998 2007 1985 1997 2009 1985 2002 2007 ...
##  $ Outlet_Size              : Factor w/ 4 levels "","High","Medium",..: 3 1 1 1 3 4 3 3 1 1 ...
##  $ Outlet_Location_Type     : Factor w/ 3 levels "Tier 1","Tier 2",..: 1 2 3 2 3 1 3 3 2 2 ...
##  $ Outlet_Type              : Factor w/ 4 levels "Grocery Store",..: 2 2 1 2 4 2 3 4 2 2 ...
```

NOTE: Item_Outlet_Sales is present in train but not in test dataset because this is the target variable that we have to predict.

Now, we combine our train and test dataset so that we don't need to do the data cleansing and data manipulation steps twice. After making the desired changes we can split the data again before doing the regression analysis.

```r
test$Item_Outlet_Sales <- NA
data_combined <- rbind(train, test)
dim(data_combined)
```

```
## [1] 14204    12
```

Univariate Analysis

Now, we find out the number of numerical predictors in our dataset.

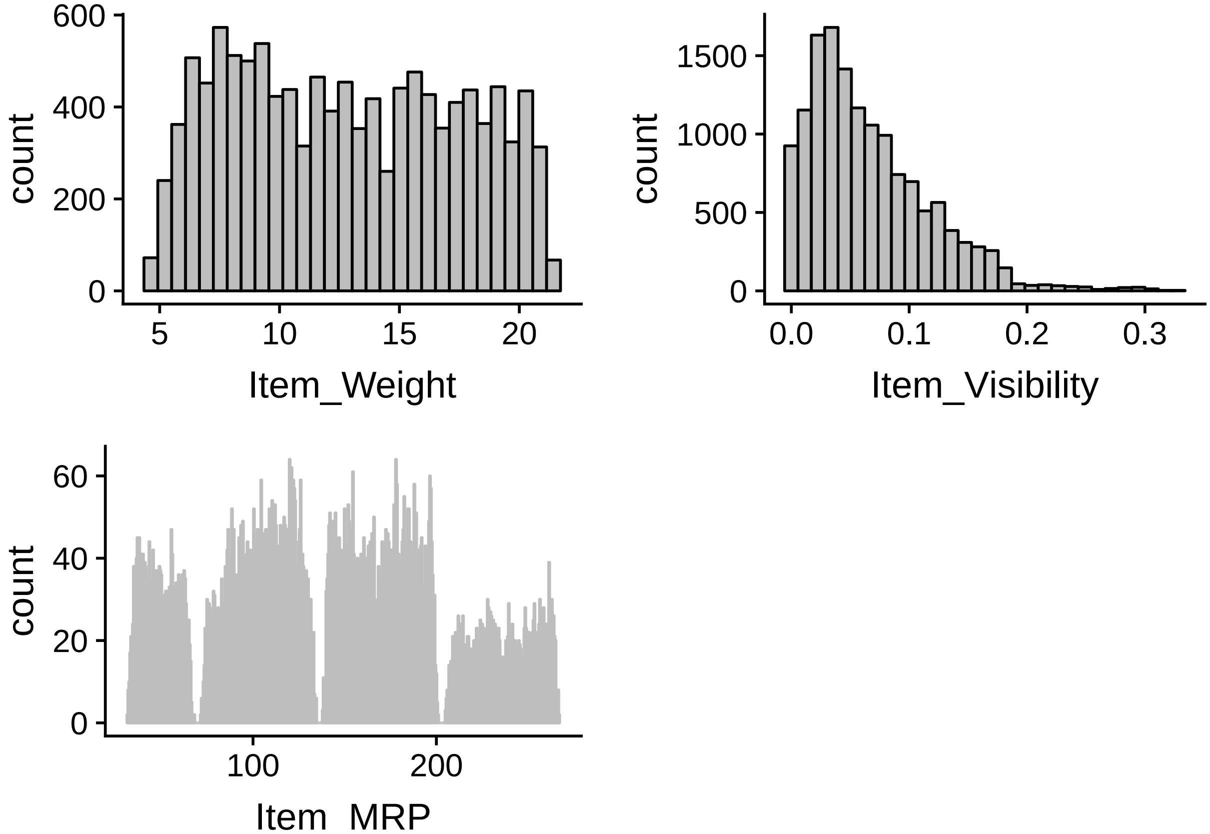
```r
train_numeric = dplyr::select_if(train, is.numeric)
names(train_numeric)
```

```
## [1] "Item_Weight"               "Item_Visibility"          
## [3] "Item_MRP"                  "Outlet_Establishment_Year"
## [5] "Item_Outlet_Sales"
```

Only three of our predictors are numeric, which implies that we have to do one-hot-coding of many of our predictors (Outlet_Establishment_Year is actually a categorical variable). We will use the histograms for visualizations because that will help us in visualizing the distribution of the variables.

```r
plot_weight <- ggplot(data_combined) + geom_histogram(aes(Item_Weight), color="black", fill="grey")
plot_visibility <- ggplot(data_combined) + geom_histogram(aes(Item_Visibility), color="black", fill="gr
plot_mrp <- ggplot(data_combined) + geom_histogram(aes(Item_MRP), color="grey", fill="grey", binwidth =
grid.arrange(plot_weight, plot_visibility, plot_mrp, ncol = 2, nrow = 2)
```

```
## Warning: Removed 2439 rows containing non-finite values (stat_bin).
```

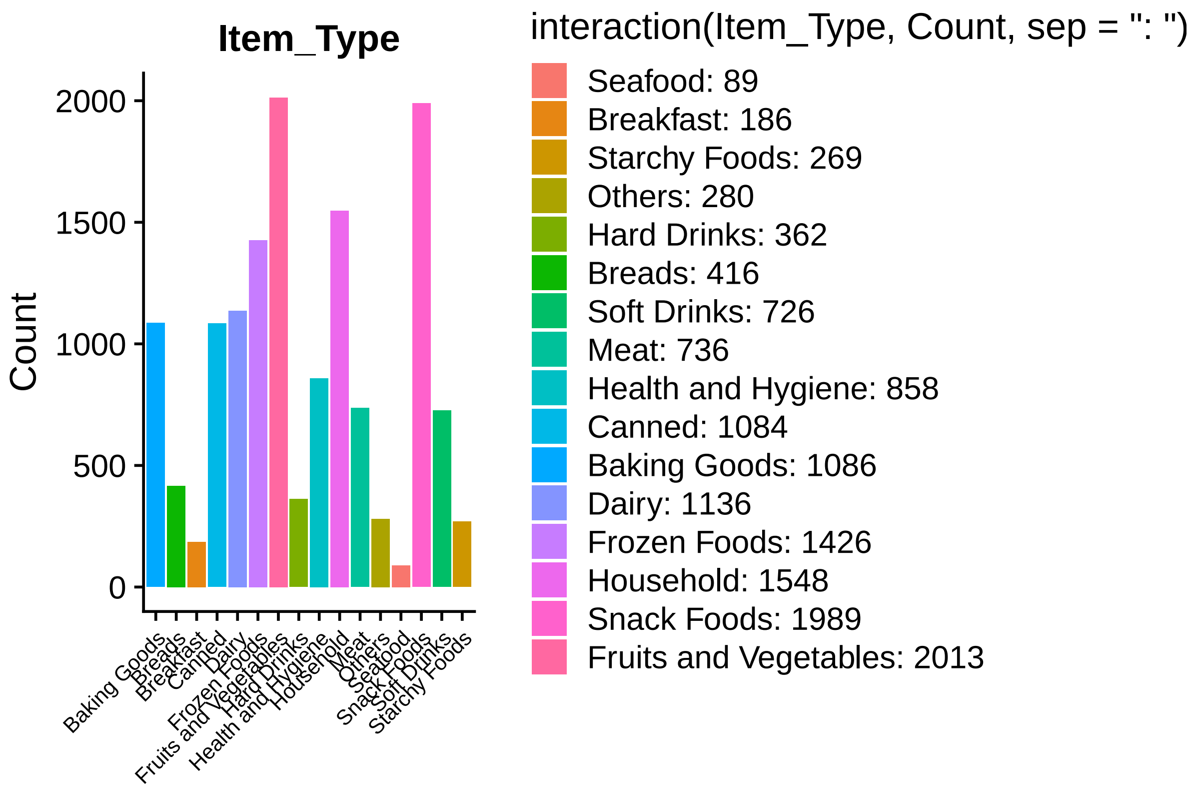


Observations:

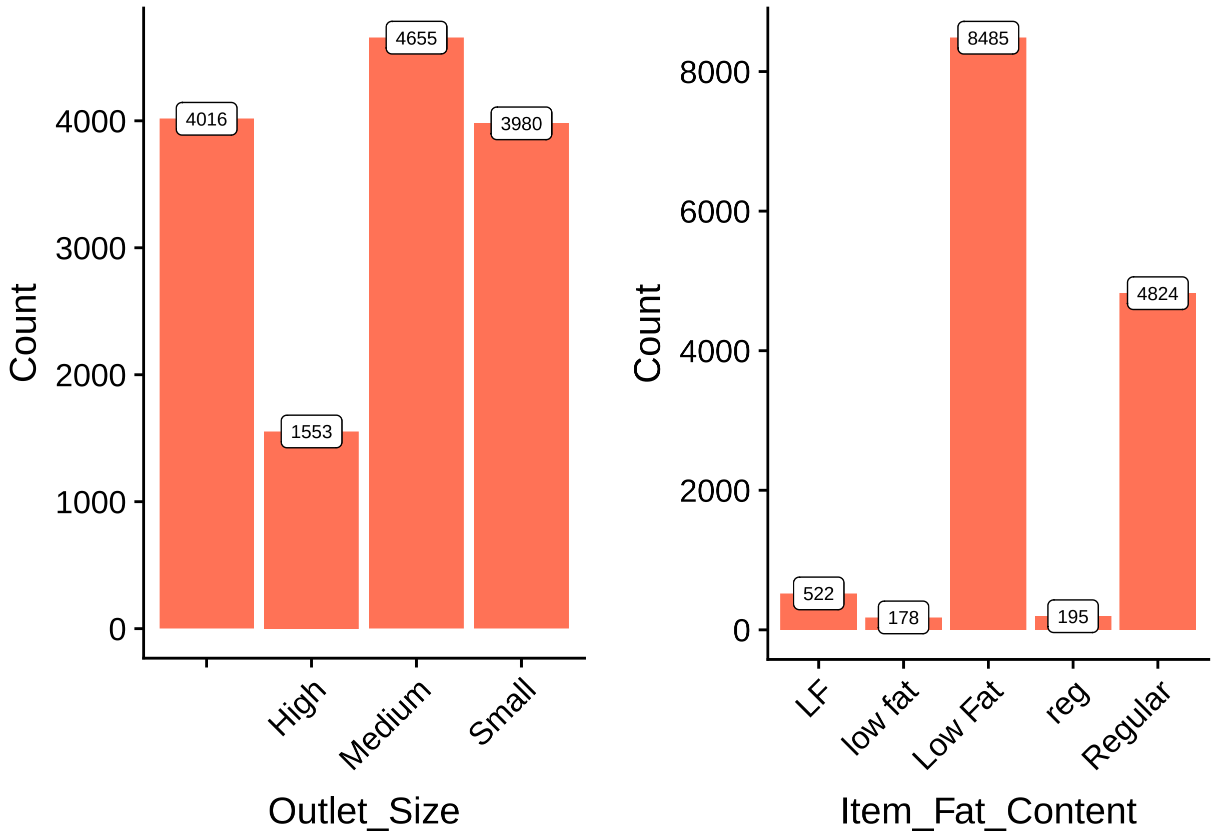
- There seems to be no clear-cut pattern in Item_Weight.
- Item_Visibility is right-skewed and should be transformed to curb its skewness.
- We can clearly see 4 different distributions for Item_MRP. It is an interesting insight.

Now, we will explore the categorical variables to gain some more insights about our dataset.

```
ggplot(data_combined %>% group_by(Item_Type) %>% summarise(Count = n())) +   geom_bar(aes(Item_Type, Co
```




```
plot_outletSize <- ggplot(data_combined %>% group_by(Outlet_Size) %>% summarise(Count = n())) +   geom_

plot_fatContent <- ggplot(data_combined %>% group_by(Item_Fat_Content) %>% summarise(Count = n())) +   g

grid.arrange(plot_outletSize, plot_fatContent, ncol = 2)
```



Observations:

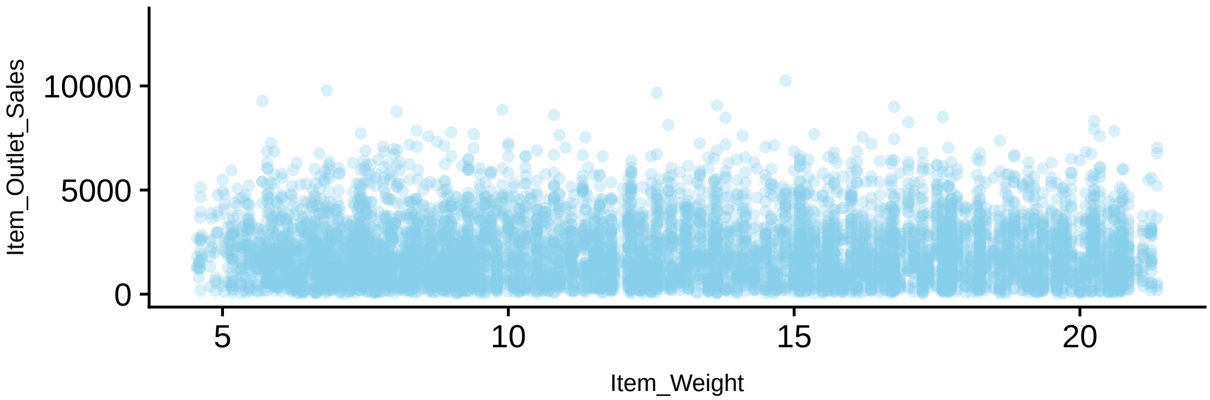
- 'LF', 'low fat', and 'Low Fat' are the same category and can be combined into one. Similarly we can combine 'reg' and 'Regular' into one.
- In Outlet_Size's plot, for 4016 observations, Outlet_Size is blank or missing.
- Some of the items are non-food items, but all the items are categorized either as low fat or regular, which is incorrect. Therefore, we need to assign separate category to non-food items.

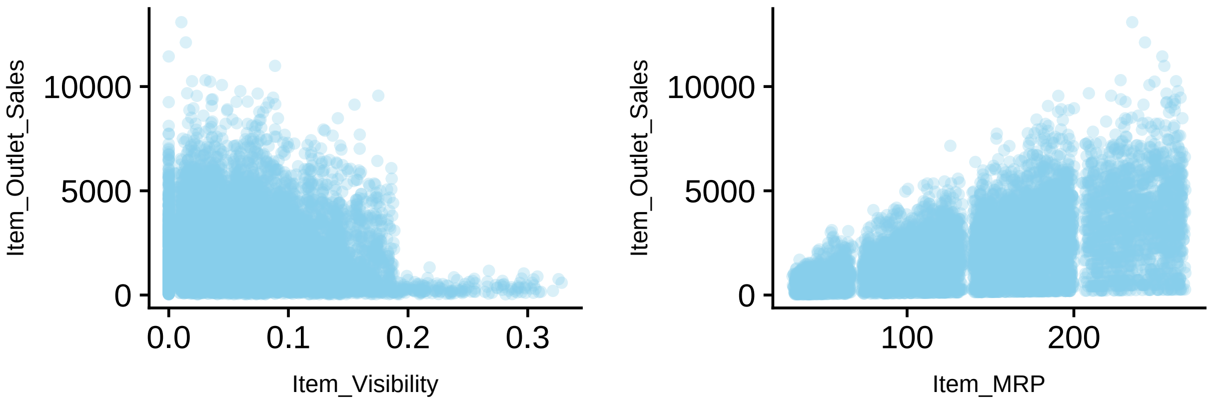
Multivariate Analysis

Now, we wll explore the independent variables with respect to the target variable. The objective is to discover hidden relationships between the independent variable and the target variable.

```
plot1 <- ggplot(train) + geom_point(aes(Item_Weight, Item_Outlet_Sales), colour = "skyblue", alpha = 0.

plot2 <- ggplot(train) + geom_point(aes(Item_Visibility, Item_Outlet_Sales), colour = "skyblue", alpha

plot3 <- ggplot(train) + geom_point(aes(Item_MRP, Item_Outlet_Sales), colour = "skyblue", alpha = 0.3)

second_row_2 = plot_grid(plot2, plot3, ncol = 2)
plot_grid(plot1, second_row_2, nrow = 2)
```

```
## Warning: Removed 1463 rows containing missing values (geom_point).
```

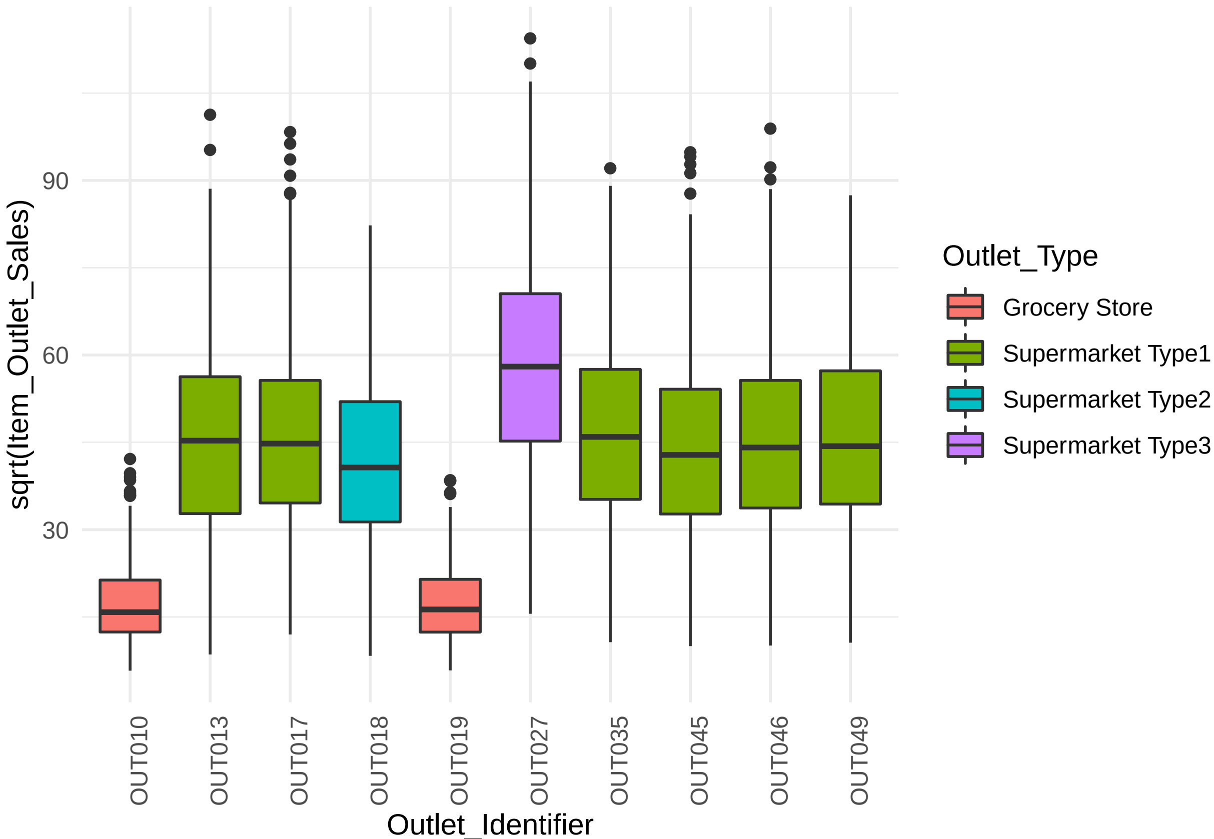




Observations:

- No obvious pattern emerges in Item_Weight vs Item_Outlet_Sales plot.
- Item_Visibility vs Item_Outlet_Sales indicates that the more visible a product is the less higher its sales will be. This might be due to the fact that a great number of daily use products, which do not need high visibility, control the top of the sales chart. Furthermore, there is a concerning number of products with visibility zero.
- In the third plot of Item_MRP vs Item_Outlet_Sales, we can clearly see 4 segments of prices that can be used in feature engineering to create a new variable.

```
plot4 <- ggplot(train) + geom_boxplot(aes(Outlet_Identifier, sqrt(Item_Outlet_Sales), fill = Outlet_Typ
plot4
```



This box plot shows the relationship between Item Outlet Sales and Outlet Identifier. The items are more frequently bought as outlet size grows, therefore we see that both OUT010 and OUT019 belongs to the grocery store category.

Stage 3: Data Pre-processing

Now, we make certain changes to our dataset from the observations we gather through exploratory data analysis. First, we combine different Item_Fat_Content categories into just two categories.

```
data_combined$Item_Fat_Content <-str_replace(str_replace(str_replace(data_combined$Item_Fat_Content,"LF

table(data_combined$Item_Fat_Content)
```

```
##
## Low Fat Regular
##    9185    5019
```

We saw on exploring part, Item_Weight column has null values which can affect the result of anaylsis. Impute missing value by median. We are using median because it is known to be highly robust to outliers. Moreover, for this problem, our evaluation metric is RMSE which is also highly affected by outliers. Hence, median is better in this case.

```
sum(is.na(data_combined$Item_Weight))
```

```
## [1] 2439
```

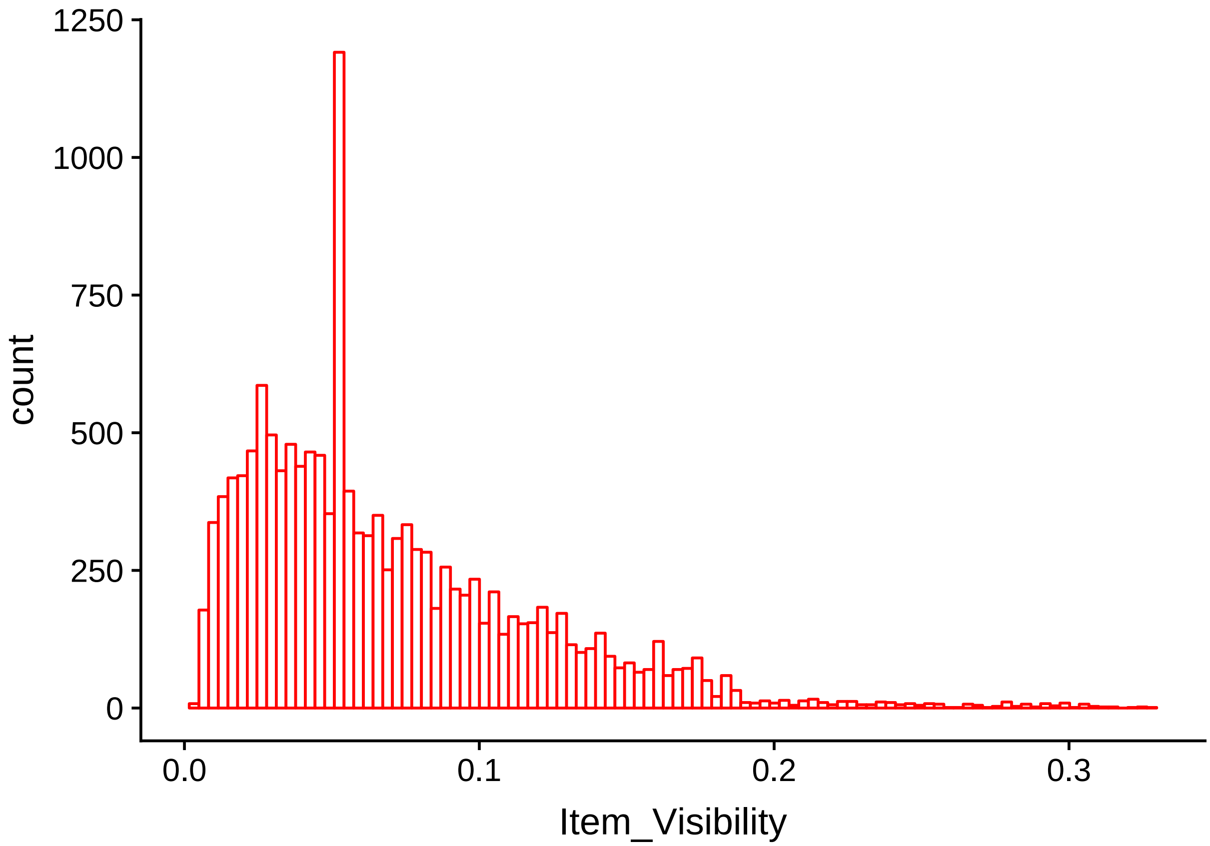
```
data_combined$Item_Weight[is.na(data_combined$Item_Weight)] <-
  median(data_combined$Item_Weight, na.rm = TRUE)
```

```r
sum(is.na(data_combined$Item_Weight))
```

```
## [1] 0
```

Let's take up Item_Visibility. On exploration part above, we saw item visibility has zero value also,which is practically not possible. Hence, we'll consider it as a missing value and once again make the imputation using median.

```r
data_combined$Item_Visibility <- ifelse(data_combined$Item_Visibility == 0,
                                        median(data_combined$Item_Visibility),
                                        data_combined$Item_Visibility)
ggplot(data_combined) + geom_histogram(aes(Item_Visibility), bins = 100, color="red", fill="white")
```



In the histogram, we can see that the issue of zero item visibility has been resolved.

We need to mutate new columns for more meaningful data. First, we evaluate Item_Identifier column because we discovered Item_Identifier column has special codes to recognize the type of item when we tried to understand data. We will use first two letters (DR = Drink, FD = Food, NC = Non-Consumable). Secondly, we generate new Outlet_Age column from Outlet_Establishment_Year. And, we will also change the values of Item_Fat_Content wherever Item_category is 'NC' because non-consumable items cannot have any fat content.

```r
data_combined <- data_combined %>%
  mutate(Item_Category = substr(Item_Identifier, 1, 2),
         Outlet_Age = 2013 - Outlet_Establishment_Year)

table(data_combined$Item_Category)
```

```
##
```

```
##    DR    FD    NC
##  1317 10201  2686
```

```r
data_combined$Item_Fat_Content[data_combined$Item_Category == "NC"] = "Non-Edible"

table(data_combined$Item_Fat_Content)
```

```
##
##    Low Fat Non-Edible    Regular
##       6499       2686       5019
```

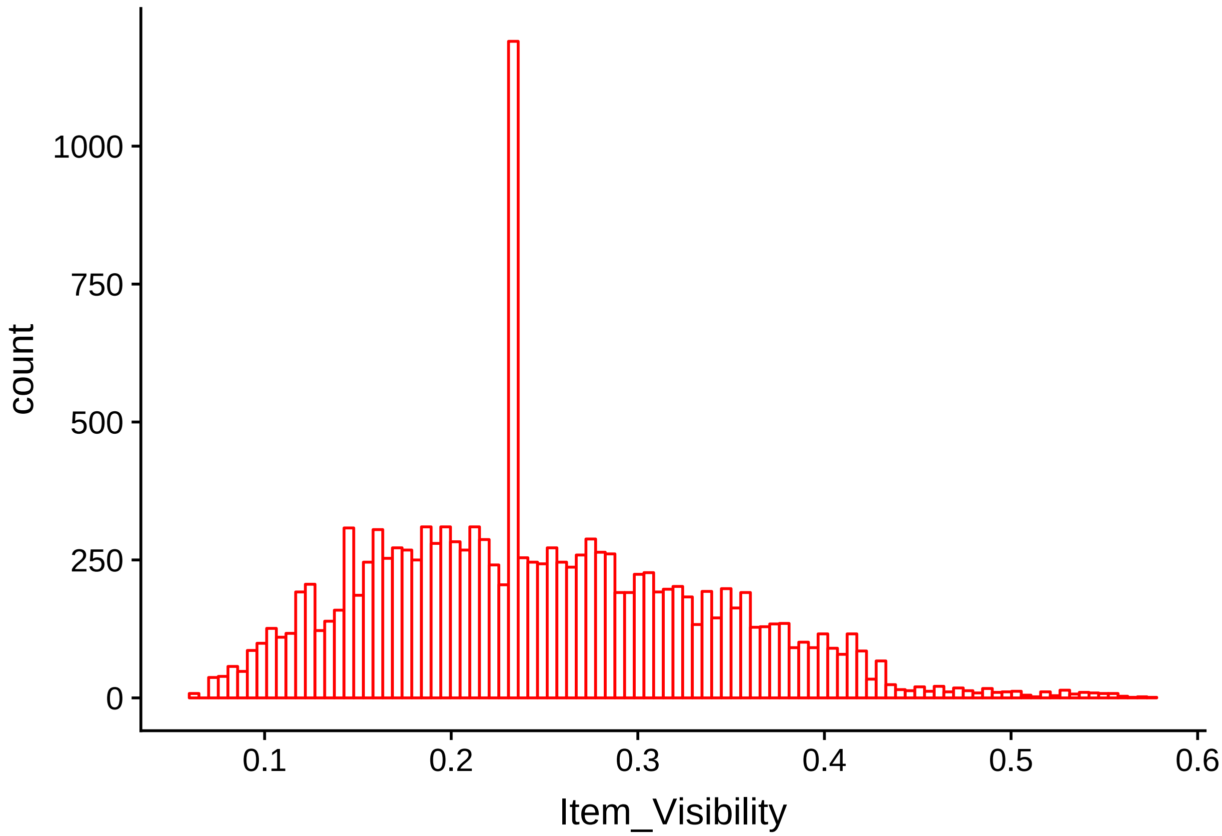
Since, machine learning and data science algorithms work only on numerical variables, we need to do One Hot Encoding for our categorical data.

```r
data_combined <- dummy.data.frame(data_combined, names = c('Item_Fat_Content', 'Outlet_Size', 'Outlet_L

data_combined <- subset(data_combined, select = -c(Item_Identifier, Item_Type, Outlet_Establishment_Yea

str(data_combined)
```

```
## 'data.frame':    14204 obs. of  32 variables:
##  $ Item_Weight                 : num  9.3 5.92 17.5 19.2 8.93 ...
##  $ Item_Fat_Content_Low Fat    : int  1 0 1 0 0 0 0 1 0 0 ...
##  $ Item_Fat_Content_Non-Edible : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ Item_Fat_Content_Regular    : int  0 1 0 1 0 1 1 0 1 1 ...
##  $ Item_Visibility             : num  0.016 0.0193 0.0168 0.054 0.054 ...
##  $ Item_MRP                    : num  249.8 48.3 141.6 182.1 53.9 ...
##  $ Outlet_Identifier_OUT010    : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ Outlet_Identifier_OUT013    : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ Outlet_Identifier_OUT017    : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Outlet_Identifier_OUT018    : int  0 1 0 0 0 1 0 0 0 0 ...
##  $ Outlet_Identifier_OUT019    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Identifier_OUT027    : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Outlet_Identifier_OUT035    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Identifier_OUT045    : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ Outlet_Identifier_OUT046    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Identifier_OUT049    : int  1 0 1 0 0 0 0 0 0 0 ...
##  $ Outlet_Size_                : int  0 0 0 1 0 0 0 0 1 1 ...
##  $ Outlet_Size_High            : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ Outlet_Size_Medium          : int  1 1 1 0 0 1 0 1 0 0 ...
##  $ Outlet_Size_Small           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Location_Type_Tier 1  : int  1 0 1 0 0 0 0 0 0 0 ...
##  $ Outlet_Location_Type_Tier 2  : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ Outlet_Location_Type_Tier 3  : int  0 1 0 1 1 1 1 1 0 0 ...
##  $ Outlet_Type_Grocery Store    : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ Outlet_Type_Supermarket Type1: int  1 0 1 0 1 0 1 0 1 1 ...
##  $ Outlet_Type_Supermarket Type2: int  0 1 0 0 0 1 0 0 0 0 ...
##  $ Outlet_Type_Supermarket Type3: int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Item_Outlet_Sales           : num  3735 443 2097 732 995 ...
##  $ Item_Category_DR            : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ Item_Category_FD            : int  1 0 1 1 0 1 1 1 1 1 ...
##  $ Item_Category_NC            : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ Outlet_Age                  : num  14 4 14 15 26 4 26 28 11 6 ...
```

Now, remove skewness from the variable Item_Visibility.

```r
data_combined$Item_Visibility <- sqrt(data_combined$Item_Visibility)
ggplot(data_combined) + geom_histogram(aes(Item_Visibility), bins = 100, color="red", fill="white")
```



Now, we scale our numerical predeictors. We use Z-score normalization.

```r
data_combined$Item_Weight <- scale(data_combined$Item_Weight, center= TRUE, scale=TRUE)
data_combined$Item_Visibility <- scale(data_combined$Item_Visibility, center= TRUE, scale=TRUE)
data_combined$Item_MRP <- scale(data_combined$Item_MRP, center= TRUE, scale=TRUE)
data_combined$Outlet_Age <- scale(data_combined$Outlet_Age, center= TRUE, scale=TRUE)

str(data_combined)
```

```
## 'data.frame':    14204 obs. of  32 variables:
##  $ Item_Weight                  : num [1:14204, 1] -0.817 -1.615 1.119 1.521 -0.904 ...
##   ..- attr(*, "scaled:center")= num 12.8
##   ..- attr(*, "scaled:scale")= num 4.23
##  $ Item_Fat_Content_Low Fat     : int  1 0 1 0 0 0 0 1 0 0 ...
##  $ Item_Fat_Content_Non-Edible  : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ Item_Fat_Content_Regular     : int  0 1 0 1 0 1 1 0 1 1 ...
##  $ Item_Visibility              : num [1:14204, 1] -1.366 -1.228 -1.334 -0.174 -0.174 ...
##   ..- attr(*, "scaled:center")= num 0.248
##   ..- attr(*, "scaled:scale")= num 0.0887
##  $ Item_MRP                     : num [1:14204, 1] 1.75245 -1.49364 0.00987 0.66181 -1.40357 ...
##   ..- attr(*, "scaled:center")= num 141
##   ..- attr(*, "scaled:scale")= num 62.1
##  $ Outlet_Identifier_OUT010     : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ Outlet_Identifier_OUT013     : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ Outlet_Identifier_OUT017     : int  0 0 0 0 0 0 0 0 0 1 ...
```

```
##  $ Outlet_Identifier_OUT018     : int  0 1 0 0 0 1 0 0 0 0 ...
##  $ Outlet_Identifier_OUT019     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Identifier_OUT027     : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Outlet_Identifier_OUT035     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Identifier_OUT045     : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ Outlet_Identifier_OUT046     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Identifier_OUT049     : int  1 0 1 0 0 0 0 0 0 0 ...
##  $ Outlet_Size_                 : int  0 0 0 1 0 0 0 0 1 1 ...
##  $ Outlet_Size_High             : int  0 0 0 0 1 0 1 0 0 0 ...
##  $ Outlet_Size_Medium           : int  1 1 1 0 0 1 0 1 0 0 ...
##  $ Outlet_Size_Small            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Outlet_Location_Type_Tier 1  : int  1 0 1 0 0 0 0 0 0 0 ...
##  $ Outlet_Location_Type_Tier 2  : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ Outlet_Location_Type_Tier 3  : int  0 1 0 1 1 1 1 1 0 0 ...
##  $ Outlet_Type_Grocery Store    : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ Outlet_Type_Supermarket Type1: int  1 0 1 0 1 0 1 0 1 1 ...
##  $ Outlet_Type_Supermarket Type2: int  0 1 0 0 0 1 0 0 0 0 ...
##  $ Outlet_Type_Supermarket Type3: int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Item_Outlet_Sales            : num  3735 443 2097 732 995 ...
##  $ Item_Category_DR             : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ Item_Category_FD             : int  1 0 1 1 0 1 1 1 1 1 ...
##  $ Item_Category_NC             : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ Outlet_Age                   : num [1:14204, 1] -0.1397 -1.3342 -0.1397 -0.0202 1.2937 ...
##   ..- attr(*, "scaled:center")= num 15.2
##   ..- attr(*, "scaled:scale")= num 8.37
```

Now, we split the data_combined back into train and test data for building our model.

```r
train <- data_combined[1:nrow(train), ]
test <- data_combined[(nrow(train) + 1):nrow(data_combined), ]
test <- subset(test, select = -c(Item_Outlet_Sales))

dim(train)
```

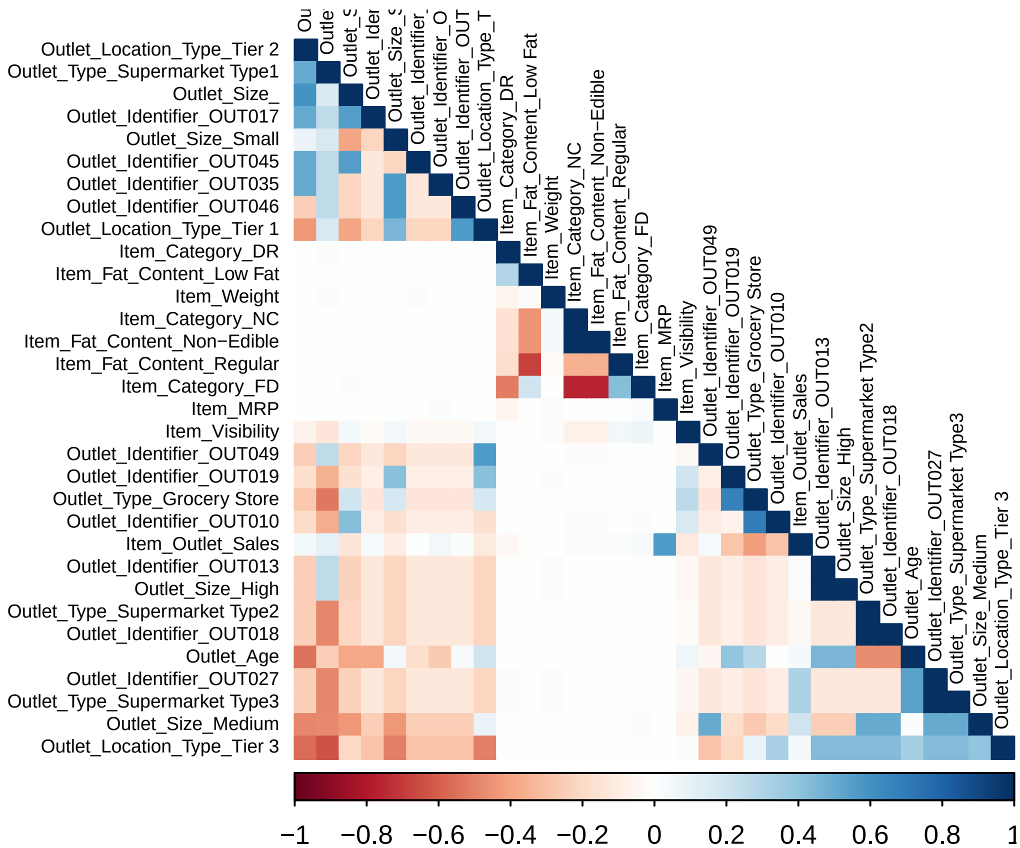
```
## [1] 8523   32
```

```r
dim(test)
```

```
## [1] 5681   31
```

Now, we check for the correlation among the variables to decide whether we need to reduce dimensions of our dataset or not.

```r
corMatrix <- cor(train[, -35])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.6, tl.col = 'black')
```



Positive correlations are displayed in blue and negative correlations in red color. Since, there is insignificant correlation (less large intensity color blocks) among the variables, we don't need to perform principal component analysis on our dataset.

Stage 4: Model Building

We will use different models ranging from simple linear models to complex models like RandomForest and XGBoost for our regression analysis. We also need to evaluate our model's performance. For that we will use **Root Mean Squared Error (RMSE)**. It measures the square root of the average of the squared difference between the predictions and the ground truth. Since the RMSE is squaring the difference between the predictions and the ground truth, any significant difference is made more substantial when it is being squared. Moreover, RMSE is more sensitive to outliers.

Linear Regression It is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). *Source:* Wikipedia

```r
# building the model
set.seed(1000)
linear_reg_mod = lm(Item_Outlet_Sales ~ ., data = train)
summary(linear_reg_mod)
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4312.5  -677.9   -88.8   571.2  7929.7
```

```
##
## Coefficients: (16 not defined because of singularities)
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        2391.589     40.496  59.058  < 2e-16 ***
## Item_Weight                          -2.507     12.283  -0.204  0.83827
## `Item_Fat_Content_Low Fat`          -41.725     28.287  -1.475  0.14023
## `Item_Fat_Content_Non-Edible`       -69.970     35.115  -1.993  0.04634 *
## Item_Fat_Content_Regular                 NA         NA      NA       NA
## Item_Visibility                     -12.400     12.677  -0.978  0.32804
## Item_MRP                            965.870     12.209  79.113  < 2e-16 ***
## Outlet_Identifier_OUT010          -2006.206     61.279 -32.739  < 2e-16 ***
## Outlet_Identifier_OUT013            -66.237     52.307  -1.266  0.20544
## Outlet_Identifier_OUT017              6.394     52.390   0.122  0.90287
## Outlet_Identifier_OUT018           -373.831     52.363  -7.139 1.02e-12 ***
## Outlet_Identifier_OUT019          -1989.062     62.441 -31.855  < 2e-16 ***
## Outlet_Identifier_OUT027           1353.060     52.277  25.883  < 2e-16 ***
## Outlet_Identifier_OUT035             47.129     52.336   0.901  0.36788
## Outlet_Identifier_OUT045           -165.395     52.353  -3.159  0.00159 **
## Outlet_Identifier_OUT046            -97.341     52.336  -1.860  0.06293 .
## Outlet_Identifier_OUT049                 NA         NA      NA       NA
## Outlet_Size_                             NA         NA      NA       NA
## Outlet_Size_High                         NA         NA      NA       NA
## Outlet_Size_Medium                       NA         NA      NA       NA
## Outlet_Size_Small                        NA         NA      NA       NA
## `Outlet_Location_Type_Tier 1`            NA         NA      NA       NA
## `Outlet_Location_Type_Tier 2`            NA         NA      NA       NA
## `Outlet_Location_Type_Tier 3`            NA         NA      NA       NA
## `Outlet_Type_Grocery Store`              NA         NA      NA       NA
## `Outlet_Type_Supermarket Type1`          NA         NA      NA       NA
## `Outlet_Type_Supermarket Type2`          NA         NA      NA       NA
## `Outlet_Type_Supermarket Type3`          NA         NA      NA       NA
## Item_Category_DR                    -16.588     43.972  -0.377  0.70601
## Item_Category_FD                         NA         NA      NA       NA
## Item_Category_NC                         NA         NA      NA       NA
## Outlet_Age                               NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1128 on 8507 degrees of freedom
## Multiple R-squared:  0.5635, Adjusted R-squared:  0.5627
## F-statistic: 732.1 on 15 and 8507 DF,  p-value: < 2.2e-16
```

Just keeping the significant variables.

```r
linear_reg_mod = lm(Item_Outlet_Sales ~ Item_MRP + Outlet_Identifier_OUT010 + Outlet_Identifier_OUT018 +
summary(linear_reg_mod)
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_MRP + Outlet_Identifier_OUT010 +
##     Outlet_Identifier_OUT018 + Outlet_Identifier_OUT019 + Outlet_Identifier_OUT027 +
##     Outlet_Identifier_OUT045, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -4298.4  -678.1   -82.4   568.9  7911.6
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 2336.85      16.56 141.127  < 2e-16 ***
## Item_MRP                     966.12      12.19  79.240  < 2e-16 ***
## Outlet_Identifier_OUT010 -1993.96      50.70 -39.329  < 2e-16 ***
## Outlet_Identifier_OUT018  -351.84      40.59  -8.668  < 2e-16 ***
## Outlet_Identifier_OUT019 -1977.57      51.84 -38.144  < 2e-16 ***
## Outlet_Identifier_OUT027  1375.91      40.46  34.004  < 2e-16 ***
## Outlet_Identifier_OUT045  -143.62      40.57  -3.540 0.000402 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1129 on 8516 degrees of freedom
## Multiple R-squared:  0.5627, Adjusted R-squared:  0.5624
## F-statistic:  1826 on 6 and 8516 DF,  p-value: < 2.2e-16
```

```r
# making predictions on test data
prediction = predict(linear_reg_mod, test)
```

Score: 1200.79

Ridge Regression Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors. It is hoped that the net effect will be to give estimates that are more reliable. *For more information:* Wikepedia

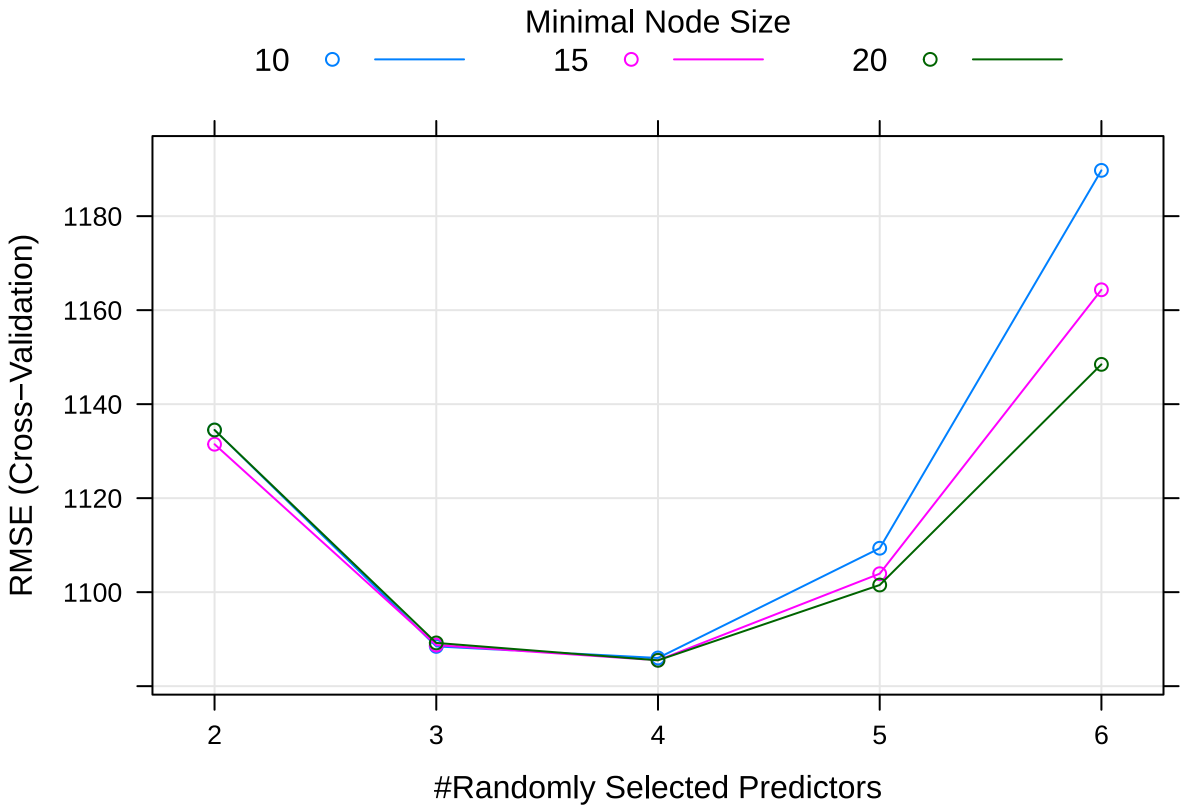
```r
set.seed(1357)
my_control <- trainControl(method="cv", number=5)
Grid <- expand.grid(alpha = 0, lambda = seq(0.001,0.1,by = 0.0002))
ridge_linear_reg_mod <- train(Item_Outlet_Sales ~ Item_MRP + Outlet_Identifier_OUT010 + Outlet_Identifi

# making predictions on test data
prediction = predict(ridge_linear_reg_mod, test)
```

Score: 1202.75

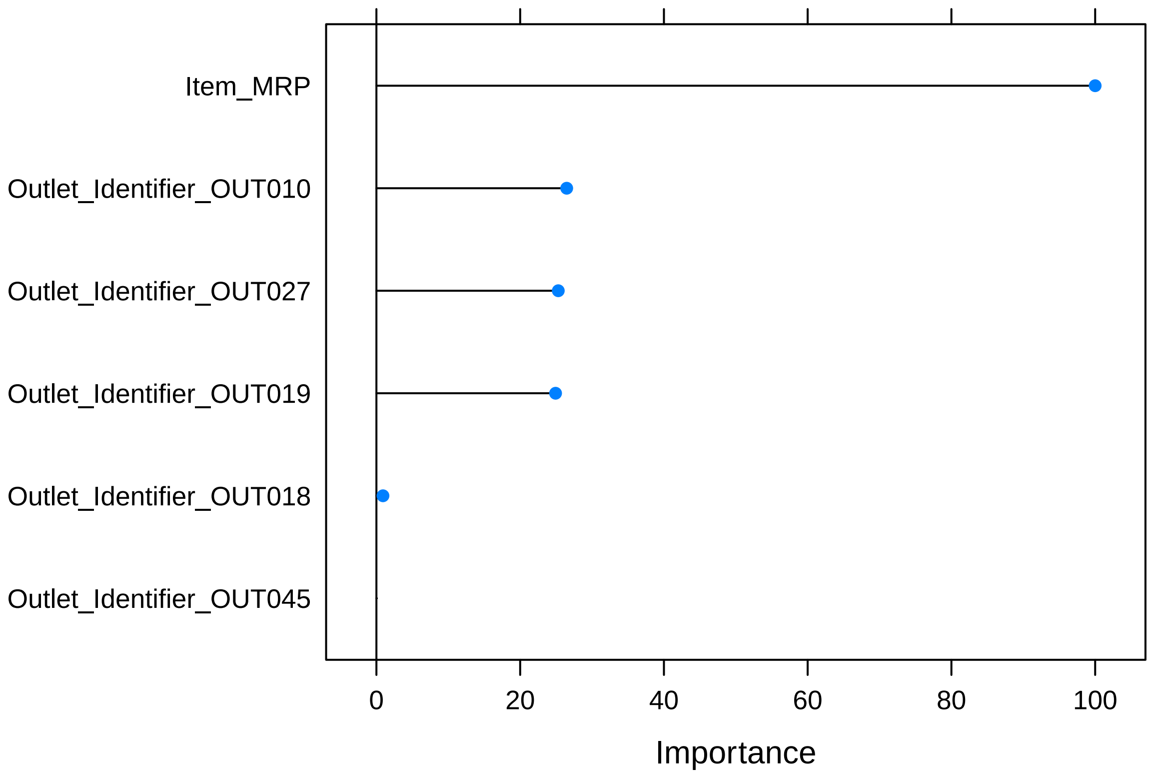
Random Forest Regression Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. *Source:* Wikipedia

```r
set.seed(1237)
my_control <- trainControl(method="cv", number=5)
tgrid = expand.grid(.mtry = c(2:6),
                    .splitrule = "variance",
                    .min.node.size = c(10,15,20))
rf_mod <- train(Item_Outlet_Sales ~ Item_MRP + Outlet_Identifier_OUT010 +
                  Outlet_Identifier_OUT018 + Outlet_Identifier_OUT019 +
                  Outlet_Identifier_OUT027 + Outlet_Identifier_OUT045, data = train,
                method='ranger',
                tuneGrid = tgrid,
                trControl= my_control,
                num.trees = 500,
```

```
                 importance = "permutation")

prediction = predict(rf_mod, test)
write.csv(prediction, "Linear_Reg_submit.csv", row.names = F)

plot(rf_mod)
```




```
plot(varImp(rf_mod))
```



Score: 1150.56

Hence, we see how we improve our regression analysis by using more robust algorithms. We can also try creating new variables from the existing variables, apply regularization techniques and use different models like neural networks to improve our results.