UNIVERSITÀ DEGLI STUDI DI MILANO

Statistical Learning, Deep Learning and Artificial Intelligence

Instructor: Respected Prof. Silvia Salini

Student: Vickysingh Chandansingh Baghel (964952) - Data Science and Economics

Project: Analysis of Big Mart Sales dataset using Statistical methods .

Index

Abstract:

Estimating future sales is the major aspect of the numerous distributions, manufacturing, marketing and wholesaling companies involved. This helps businesses to allocate capital effectively, to forecast realistic sales revenues as well as to prepare a better plan for potentially increasing the business. In this paper, estimating product sale from a single outlet is carried out using a random forest regression approach, XG booster approach which provides better predictive
results compared to a linear regression model. This approach is carried out on data from Big-Mart Sales where data discovery, processed and sufficient relevant data is extracted which play a vital role in predicting accurate outcome using R Studio.
Keywords: Machine Learning, Sales Forecasting, Random Forest, Regression, Xgboost.

1. Introduction

Estimating future sales is an important aspect of any business. Accurate prediction of future sales help companies to develop and improve business strategies as well as to gain proper market knowledge. Standard sales forecast helps companies to analyze the situation which has occurred before and then apply customer purchase inferences to identify inadequacies and weaknesses before budgeting as well as to prepare good plan for the next year. A detailed knowledge of past opportunities permits one to plan for future market needs and increase the possibility of success Regardless of external factors, firms which see sales modeling as its first step towards improved performance compared to those who don't.

The Sales prediction is very common real- life problem every company do sales prediction at least once in life time. If done correctly it can give the significant impact on performance of the company. According to many study companies with accurate sales predictions are 10% more likely to grow their revenues Y-O-Y & 7.3% more likely to hits quota.

The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined.

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

2. Methodology:

For building a model to predict accurate results the dataset of Big Mart sales undergoes several sequence of steps as mentioned in Figure 1 and in this work we propose a model using Xgboost technique. Every step plays a vital role for build-Ing the proposed model. In our model we have used 2013 Big mart dataset .After preprocessing and filling missing values, we used Linear regression, Ridge regression, Random forest and Xgboost. RSME are used as accuracy metrics for predicting the sales in Big Mart. From the accuracy metrics it was found that the model will predict best using minimum RSME. The details of the proposed method is explained in the following section.

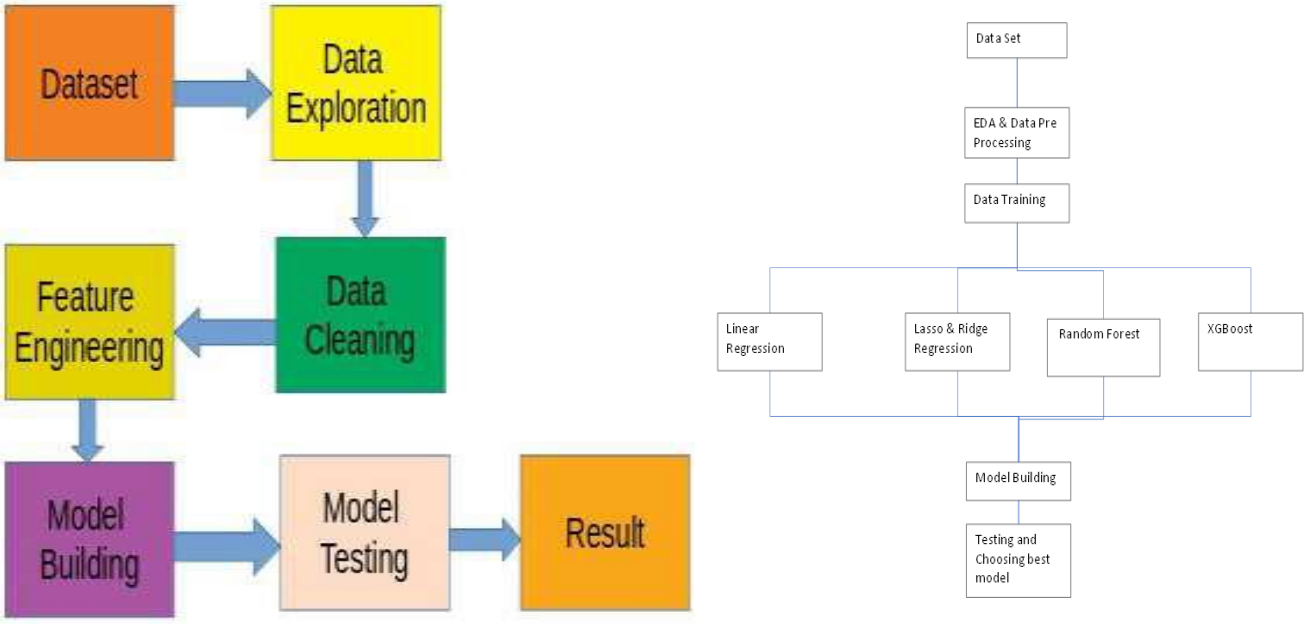


Fig: 1 Working procedure of proposed model

3. Data Collection

We collect this data from Kaggle, data is divided in 2 section train & test. Train Dataset consists of 12 variables and 8523 obs. These are divided into answer variables and predictors. Test data consist of 11 variables and 5681 obs. All the variables are answer variables. As we can see below in dataset there are numeric and categorical variables are present.

3.1 Data set overview :

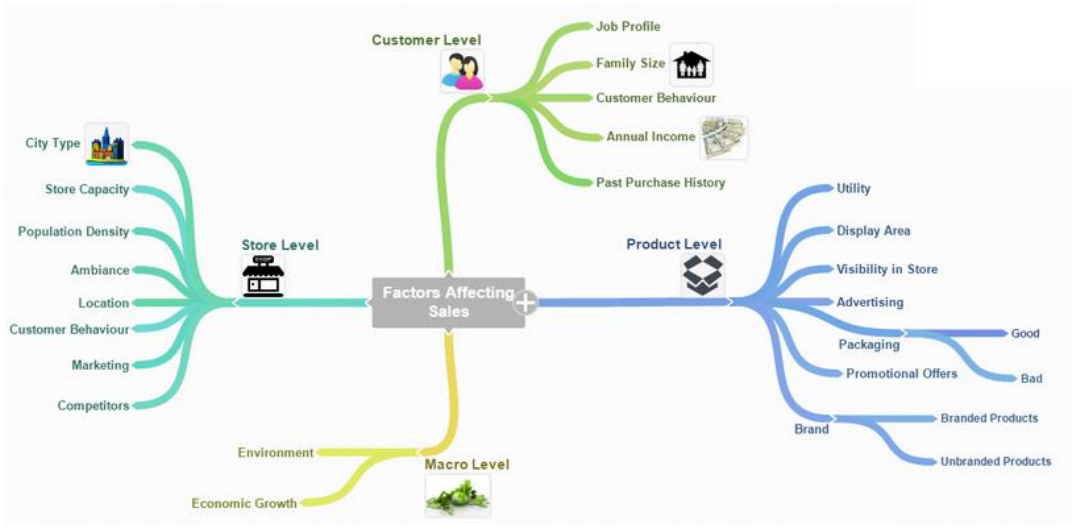
```
> dim(train)
[1] 8523   12
> dim(test)
[1] 5681   11
Classes ‘data.table’ and 'data.frame':          8523 obs. of  12 variables:
 $ Item_Identifier         : chr  "FDA15" "DRC01" "FDN15" "FDX07" ...
 $ Item_Weight             : num  9.3 5.92 17.5 19.2 8.93 ...
 $ Item_Fat_Content        : chr  "Low Fat" "Regular" "Low Fat" "Regular" ...
 $ Item_Visibility         : num  0.016 0.0193 0.0168 0 0 ...
 $ Item_Type               : chr  "Dairy" "Soft Drinks" "Meat" "Fruits and Vegetables" ...
 $ Item_MRP                : num  249.8 48.3 141.6 182.1 53.9 ...
```

```
 $ Outlet_Identifier       : chr  "OUT049" "OUT018" "OUT049" "OUT010" ...
 $ Outlet_Establishment_Year: int  1999 2009 1999 1998 1987 2009 1987 1985 2002 2007 ...
 $ Outlet_Size             : chr  "Medium" "Medium" "Medium" "" ...
 $ Outlet_Location_Type    : chr  "Tier 1" "Tier 3" "Tier 1" "Tier 3" ...
 $ Outlet_Type             : chr  "Supermarket Type1" "Supermarket Type2" "Supermarket Type1" "Grocery Store" ...
 $ Item_Outlet_Sales       : num  3735 443 2097 732 995 ...
 - attr(*, ".internal.selfref")=<externalptr>
> str(test)
Classes ‘data.table’ and 'data.frame':          5681 obs. of  12 variables:
 $ Item_Identifier         : chr  "FDW58" "FDW14" "NCN55" "FDQ58" ...
 $ Item_Weight             : num  20.75 8.3 14.6 7.32 NA ...
 $ Item_Fat_Content        : chr  "Low Fat" "reg" "Low Fat" "Low Fat" ...
 $ Item_Visibility         : num  0.00756 0.03843 0.09957 0.01539 0.1186 ...
 $ Item_Type               : chr  "Snack Foods" "Dairy" "Others" "Snack Foods" ...
 $ Item_MRP                : num  107.9 87.3 241.8 155 234.2 ...
 $ Outlet_Identifier       : chr  "OUT049" "OUT017" "OUT010" "OUT017" ...
 $ Outlet_Establishment_Year: int  1999 2007 1998 2007 1985 1997 2009 1985 2002 2007 ...
 $ Outlet_Size             : chr  "Medium" "" "" "" ...
 $ Outlet_Location_Type    : chr  "Tier 1" "Tier 2" "Tier 3" "Tier 2" ...
 $ Outlet_Type             : chr  "Supermarket Type1" "Supermarket Type1" "Grocery Store" "Supermarket Type1" ...
 - attr(*, ".internal.selfref")=<externalptr>
```

4. The Hypothese:

What is hypothesis : This is a very important step in every machine learning process. It involve understanding the problem in detail by brainstorming as many factor as possible which can impact the outcome. It is done by understanding the problem statement thoroughly and before looking at the data.

Below Is the simple mind map lets understand it.



I came up with the following hypothesis while thinking about the problem. These are just my thoughts and you can come-up with many more of these. Since we’re talking about stores and products, lets make different sets for each.

4.1 Store Level Hypotheses:

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.
3. **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
4. **Competitors:** Stores having similar establishments nearby should have less sales because of more competition.
5. **Marketing:** Stores which have a good marketing division should have higher sales as it will be able to attract customers through the right offers and advertising.
6. **Location:** Stores located within popular marketplaces should have higher sales because of better access to customers.
7. **Customer Behavior:** Stores keeping the right set of products to meet the local needs of customers will have higher sales.
8. **Ambiance:** Stores which are well-maintained and managed by polite and humble people are expected to have higher footfall and thus higher sales.

4.2 Product Level Hypotheses:

1. **Brand:** Branded products should have higher sales because of higher trust in the customer.
2. **Packaging:** Products with good packaging can attract customers and sell more.
3. **Utility:** Daily use products should have a higher tendency to sell as compared to the specific use products.
4. **Display Area:** Products which are given bigger shelves in the store are likely to catch attention first and sell more.
5. **Visibility in Store:** The location of product in a store will impact sales. Ones which are right at entrance will catch the eye of customer first rather than the ones in back.
6. **Advertising:** Better advertising of products in the store will should higher sales in most cases.
7. **Promotional Offers:** Products accompanied with attractive offers and discounts will sell more..

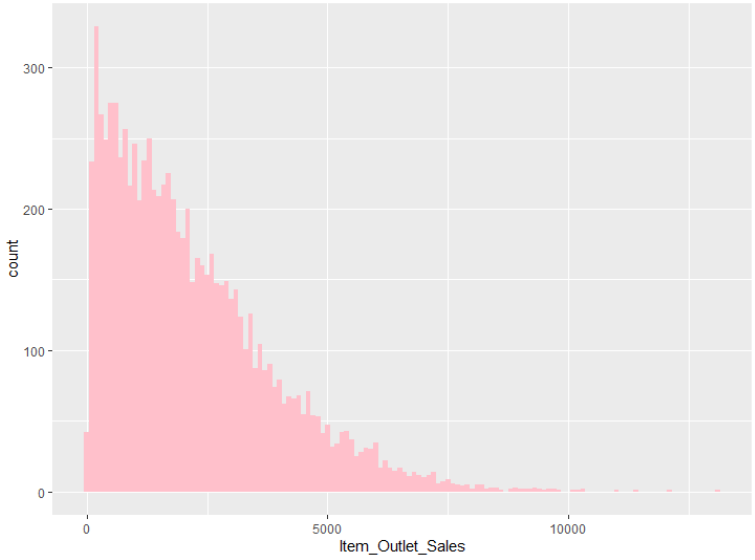
Lets move on to the data exploration where we will have a look at the data in detail.

5. Data Exploration

In this phase useful information about the data has been extracted from the dataset. That is trying to identify the information from hypotheses vs available data. Which shows that the attributes Outlet size and Item weight face the problem of missing values, also the minimum value of Item Visibility is zero which is not actually practically possible. Establishment year of Outlet varies from 1985 to 2009. These values may not be appropriate in this form. So, we need to convert them into how old a particular outlet is. There are 1559 unique products, as well as 10 unique outlets, present in the dataset. The attribute Item type contains 16 unique values. Whereas two types of Item Fat Content are there but some of them are misspelled as regular instead of 'Regular' and low fat, LF instead of Low Fat. It was found that the response variable i.e. Item Outlet Sales was positively skewed. So, to remove the skewness of response variable a log operation was performed on Item Outlet Sales

5.1 Univariate analysis:

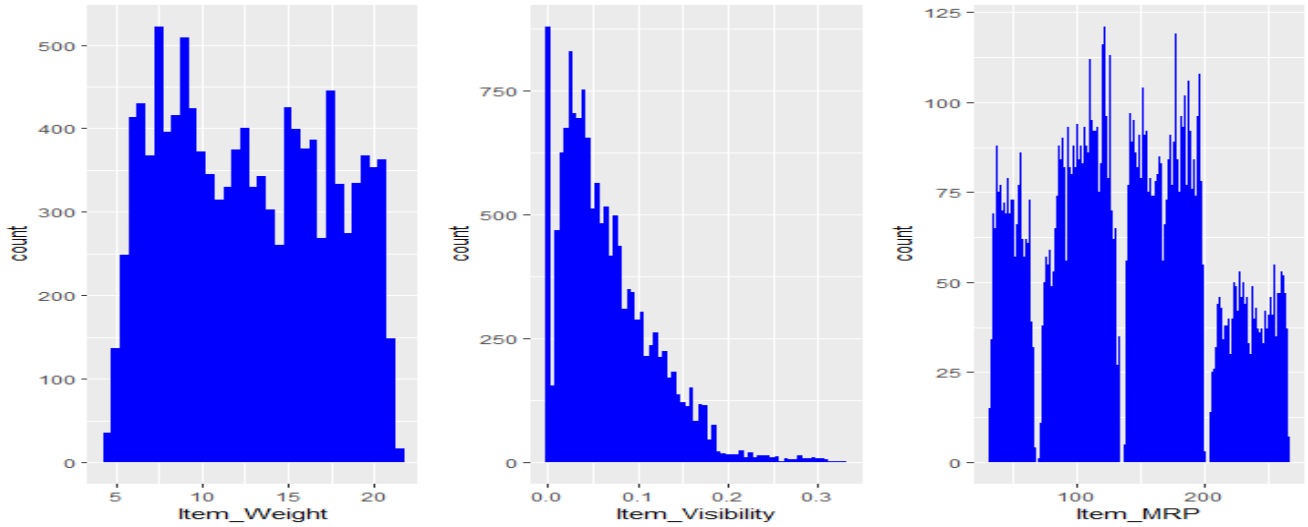
Target variable is continuous , we can visualized it plotting its hist



As you can see it is a right skewed variable & would need some data transformation to treat its skewness

5.1.1 Independent Variables(numeric variables)

Now lets check the numeric independent variables. We will again use the histogram for visualization because that will help us visualizing the distribution of the variables.



We observe:

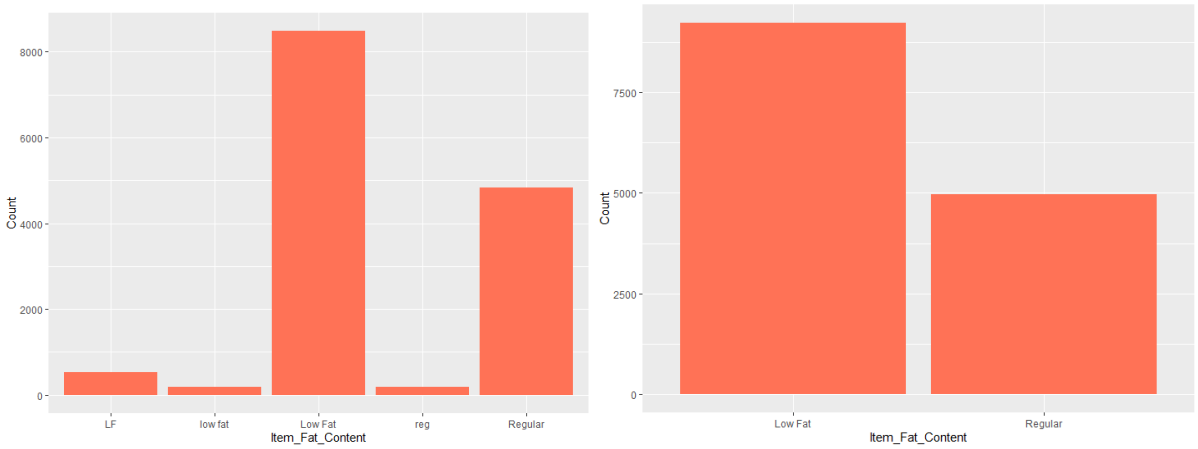
1 There seems to be no clear-cut pattern in Item_Weight.

2 Item_Visibility is right-skewed and should be transformed to curb its skewness.

3 We can clearly see 4 different distributions for Item_MRP. It is an interesting insight.

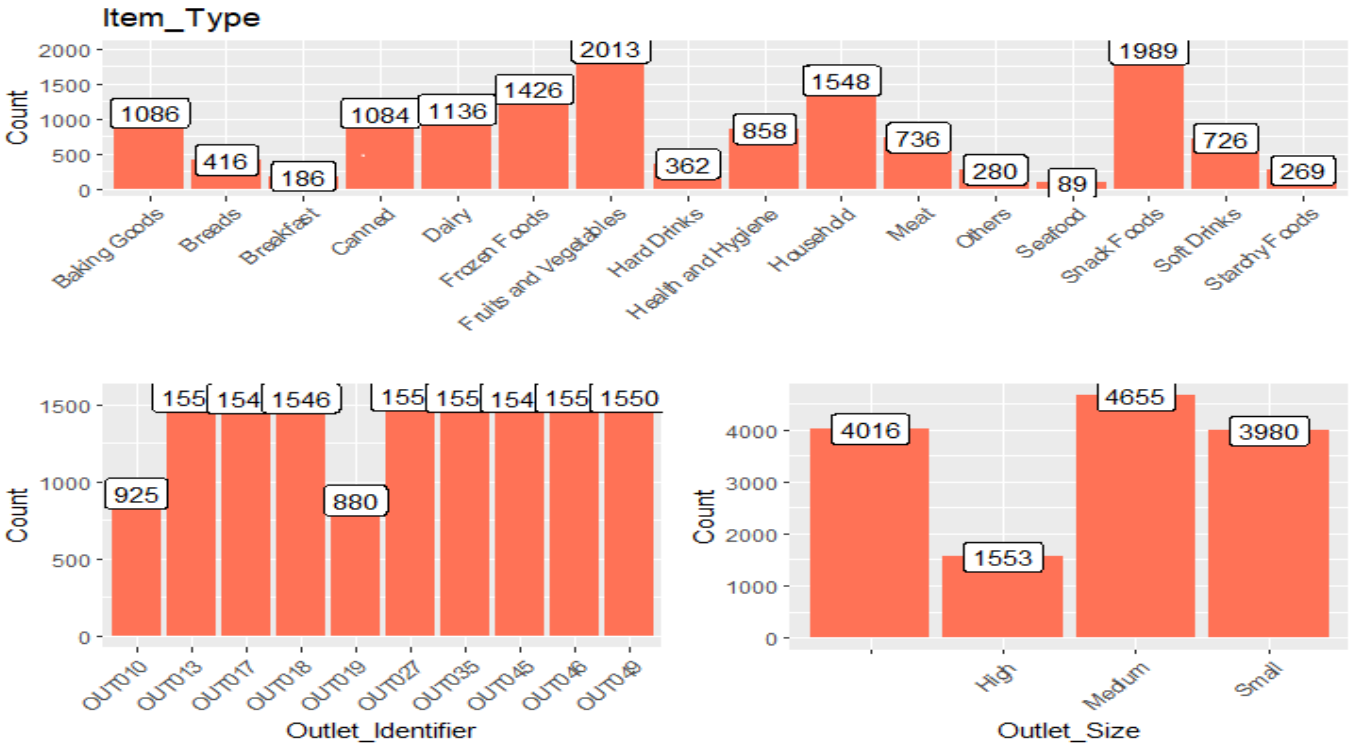
5.1.2 Independent Variables(categorical variables)

Now we will try to explore and gain some insight from the categorical variables. A categorical variables or features can have only a finite set of values . lets first plot the Item_Fat_Content.



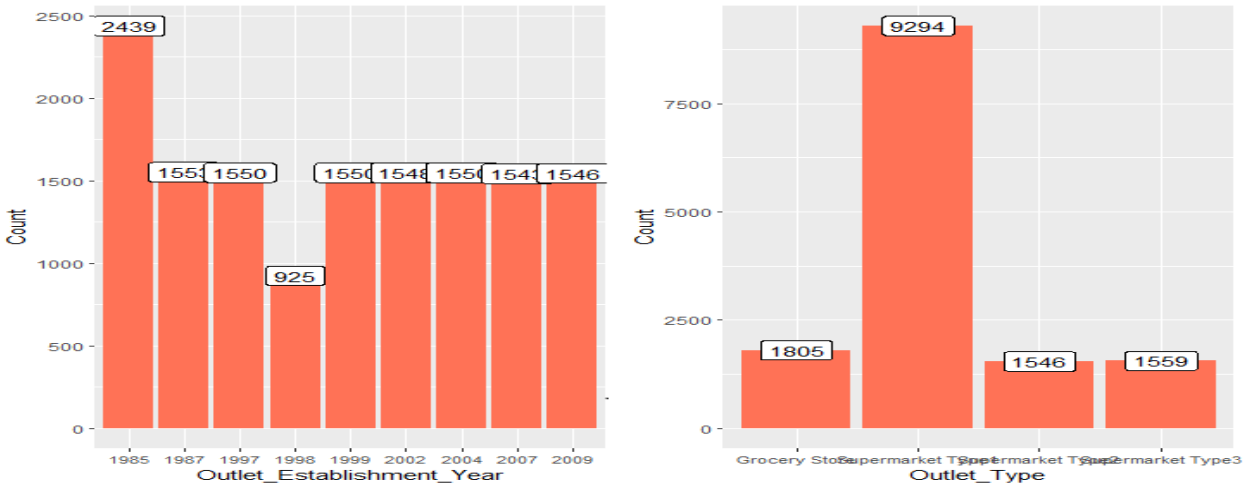
In the figure above LF,low fat & Low Fat belongs from same category and we can combine it into one. Similarly we can be done for reg & Regular into one. After making this changes we will plot same

5.1.2.1 Check the other categorical variables



In Outlet_Size plot, for 4016 obs. Outlet_Size is blank or missing. We will check for this in the bivariate analysis to substitute the missing values in the Outer_size.

5.1.2.2 The remaining categorical variable



Observations :

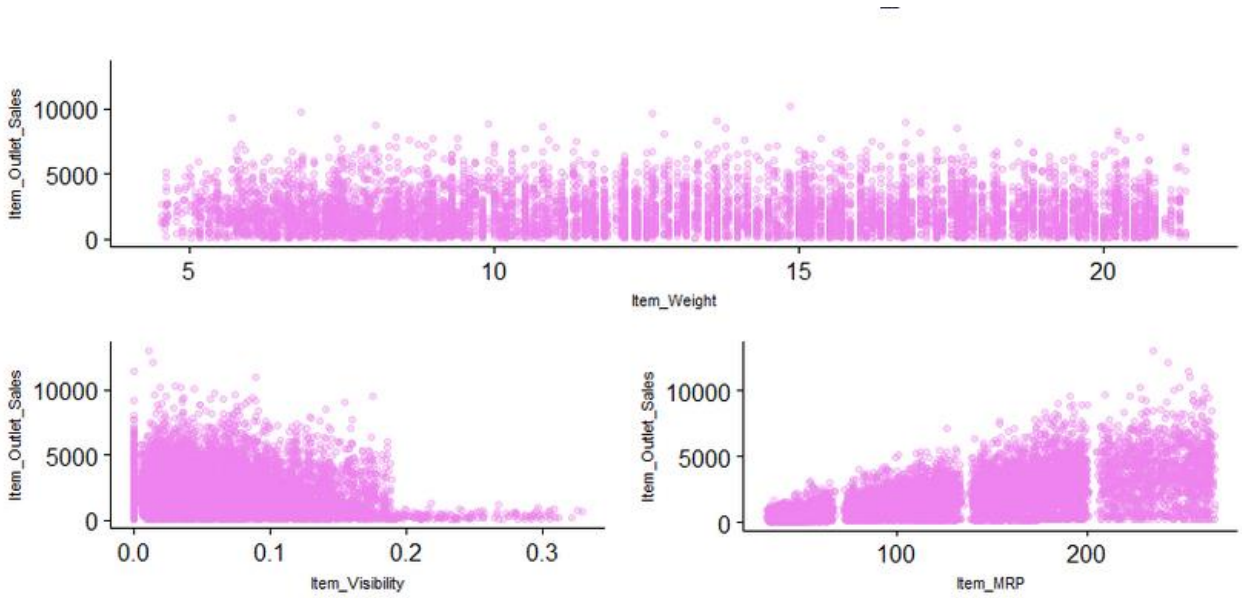
Lesser number of observation in the data for the outlier establish in the year 1998 as compared to the other years.

Supermarket Type 1 seems to be the most popular category of output

5.2 Bivariate Analysis

After looking at every feature individually, lets now do some bivariate analysis. Here we will 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 & use those finding in missing data imputation & feature engineering in next.

We will make scatter plot for continuous or numeric variables



Observation

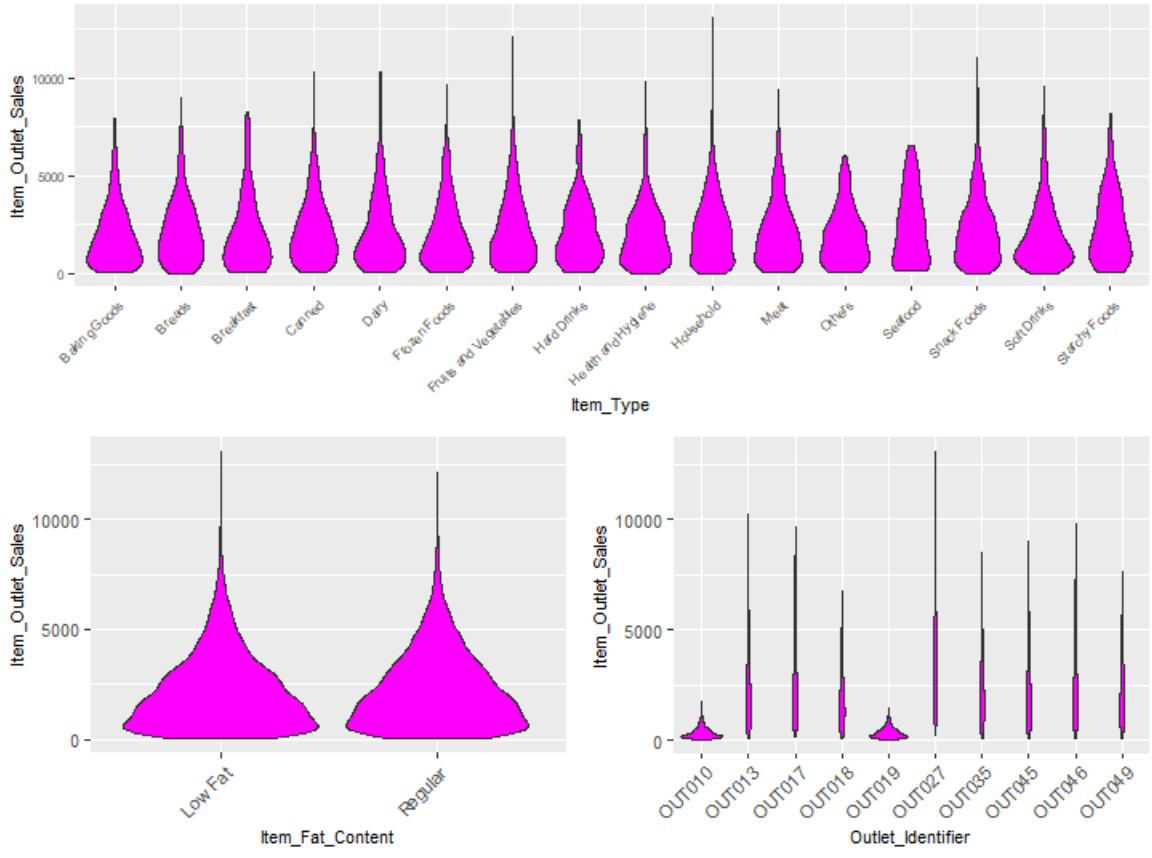
The Outlet_Sales is spread well across the entire range of the Item_Weight without any obvious pattern. In Item_Visibility Vs Outlet_Sales, there is a string of point at Item_Visibility = 0.0 which seems strange as item visibility cannot be completely zero. We will take note of this issue and deal with it in the later stages.

In the third plot of item_MRP vs Outlet_Sales, we can clearly see 4 segment of prices that can be used in feature engineering to create a new variable.

5.2.1 Target Variable Vs Independent Categorical Variables

Now will visualize the categorical variables with respect to Item_Outlet_Sales. We will try to check the distribution of the target variable across all the categories of each of the categorical variable

We could have used boxplot here but instead we will use violin plots as they show the full distribution of the data. The width of the violin plot at particular level indicates the concentration or density of the data at that level. The height of the violin tell us the range of the target variable values.

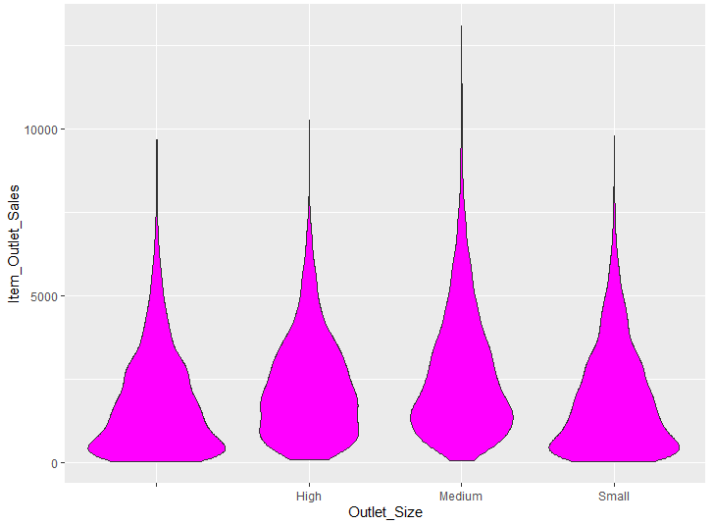


Observation

The distribution of Item_Outlet_Sales across the categories of Item_Type is not very district and same is the case with Item_Fat_Content.

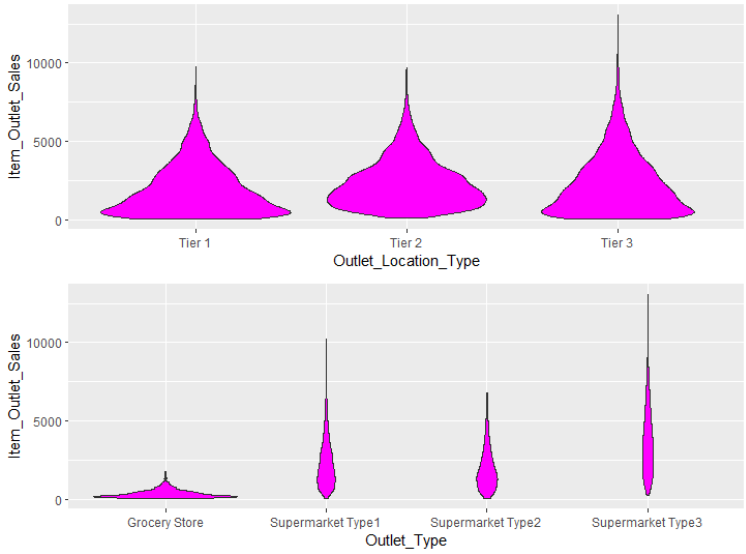
The distribution for OUT010 & OUT019 categories of Outlet_Identifier are similar and very much different from other .

In the univariate analysis we came to know about the empty values in Outlet_Size variable. Lets let check distribution of the target variable across the Outlet_Size



The distribution of small Outlet_Size is almost identical to the distribution of the blank category(first vioin) of Outlet_Size. So we can substitute the blank in Outlet_Size with small.

Lets Examine the remaining variables



Observation

Tier 1 & Tier 3 locations of the Outlet_Location_type look similar

In the Outlet_Type plot the grocery stores has most of its data points around the lower sales values as compared to the other categories.

Note: These are the kind of insight we can extract by visualizing our data. Hence, Data visualization should be an important part of any kind data analysis.

6. Data Cleaning:

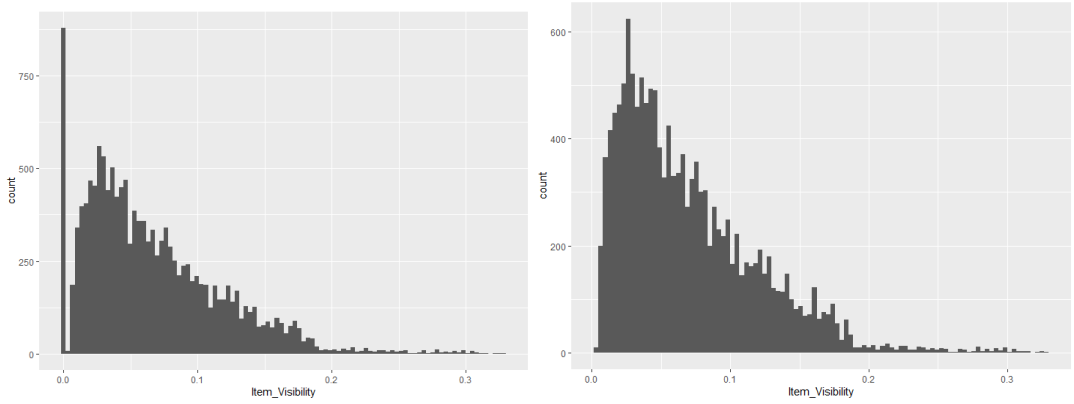
In previous section it has been found that attributes Outlet size and Element weight lack values. Here in place of missing value for outlet size, we replace with mode value of that attribute and in place of missing values of that particular attribute of object weight, we substitute by mean value. The missing attributes are numerical, where correlation between the imputed attributes decreases as well as the mean and mode replacement decreases. we believe that there is no relation among the attributes calculated and the attribute imputed in our model.

```
missing_index = which(is.na(combi$Item_Weight))
> for(i in missing_index){
+
+   item = combi$Item_Identifier[i]
+   combi$Item_Weight[i] = mean(combi$Item_Weight[combi$Item_Identifier == item], na.rm = T)
+ }
> #Cross Check
>
> sum(is.na(combi$Item_Weight))
[1] 0
```

6.1 Replacing 0 in Item_Visibility variable

Similarly, zeroes in Item_Visibility variable can be replaced with Item_Identifier wise mean values of itemVisibility. It can be visualized in the plot below.

After replacing the zeroes , we will plot histogram of Item_visibility again. In the histogram we can see that the issue of zero items visibility has been resolved.



7. Feature Engineering:

Feature engineering is all about converting cleaned data into predictive models to present the available problem in a better way. During data exploration, some noise was observed. In this phase, this noise is resolved and the data is used for building appropriate model. New features are created to make the model work precisely and effectively. A few created features can be combined for the model to work better.Feature engineering phase converts data into a form understandable by the algorithms. Most of the times, the given feature in a dataset are not sufficient to give satisfactory predictions. In such case we have to create new features which might help in improving the models performance. Lets try to create some new features for our data set

In this section we will create the following new features

Item_Types_new : Broader categories for the variable Item_Type
Item_Cotegory : Categorical variables derive from Item_Identifier
Outlet_Years : Year of operation of Outlets
Price_per_unit_wt : Item_MRP/Item_Weight
Item_MRP_cluster: Binned feature for Item_MRP

We can have a look at the item_type variable & classify the categories into perishable & non_perishable as per our understanding and make it into new feature
We did many things and create few new features you can see it in blow fig.

```
#Item_Type variable and classify the categories into perishable and non_perishable as per our understanding and make it into a new feature.

perishable = c("Breads", "Breakfast", "Dairy", "Fruits and Vegetables", "Meat", "Seafood")
non_perishable = c("Baking Goods", "Canned", "Frozen Foods", "Hard Drinks", "Health and Hygiene", "Household", "Soft Drinks")

# create a new feature 'Item_Type_new'
combi[,Item_Type_new := ifelse(Item_Type %in% perishable, "perishable", ifelse(Item_Type %in% non_perishable, "non_perishable", "not_sur
"))]

#Let's compare Item_Type with the first 2 characters of Item_Identifier, i.e., 'DR', 'FD', and 'NC'. These identifiers most probably stand f
r drinks, food, and non-consumable.

table(combi$Item_Type, substr(combi$Item_Identifier, 1, 2))

                       DR   FD   NC
Baking Goods            0 1086    0
Breads                  0  416    0
Breakfast               0  186    0
Canned                  0 1084    0
Dairy                 229  907    0
Frozen Foods            0 1426    0
Fruits and Vegetables   0 2013    0
Hard Drinks           362    0    0
Health and Hygiene      0    0  858
Household               0    0 1548
Meat                    0  736    0
Others                  0    0  280
Seafood                 0   89    0
Snack Foods             0 1989    0
Soft Drinks           726    0    0
Starchy Foods           0  269    0

# Item_category Created

combi[,Item_category := substr(combi$Item_Identifier, 1, 2)]

#Outlet_Years (years of operation) and price_per_unit_wt (price per unit weight).

combi$Item_Fat_Content[combi$Item_category == "NC"] = "Non-Edible"

# years of operation for outlets
combi[,Outlet_Years := 2013 - Outlet_Establishment_Year]
combi$Outlet_Establishment_Year = as.factor(combi$Outlet_Establishment_Year)

# Price per unit weight
combi[,price_per_unit_wt := Item_MRP/Item_Weight]

#the Item_MRP vs Item_Outlet_Sales plot, we saw Item_MRP was spread across in 4 chunks. Now let's assign a label to each of these chunks and
use this label as a new variable.

# creating new independent variable - Item_MRP_clusters
combi[,Item_MRP_clusters := ifelse(Item_MRP < 69, "1st",
                                   ifelse(Item_MRP >= 69 & Item_MRP < 136, "2nd",
                                          ifelse(Item_MRP >= 136 & Item_MRP < 203, "3rd", "4th")))]
```

8. Encoding Categorical Variables:

Most of the machine learning algorithms produce better result with numerical variables only. So, it is essential to treat the categorical variables present in the data for giving the statistical weightage to the variable
We will use 2 techniques - Label Encoding and One Hot Encoding.

1 Label encoding simply means converting each category in a variable to a number. It is more suitable for ordinal variables - categorical variables with some order.

2 In One hot encoding, each category of a categorical variable is converted into a new binary column (1/0).

9. Preprocessing Data:

In simple words, pre-processing refers to the transformations applied to your data before feeding it to the algorithm. It involves further cleaning of data, data transformation, data scaling and many more things.

For our data, we will deal with skewness and scale the numerical variables

9.1 Removing skewness:

Skewness in variable is undesirable for predictive modeling. Some machine learning algorithms assume normally distributed data and a skewed variable can be transformed by taking its log, square root, or cube root so that to make its distribution as close as normal distribution. In our data variables Item Visibility and price_per_unit_wt are highly skewed. So, we will treat the skewness with the help of log transformation.

```
skewness(combi$Item_Visibility); skewness(combi$price_per_unit_wt)
[1] 1.254046
[1] 1.304551
> #Removing Skewness
> combi[,Item_Visibility := log(Item_Visibility + 1)] # log + 1 to avoid division by zero
> combi[,price_per_unit_wt := log(price_per_unit_wt + 1)]
```

9.2 Scaling numeric predictors:

Let's scale and center the numeric variables to make them have a mean of zero, standard deviation of one and scale of 0 to 1. Scaling and centering is required for linear regression models.

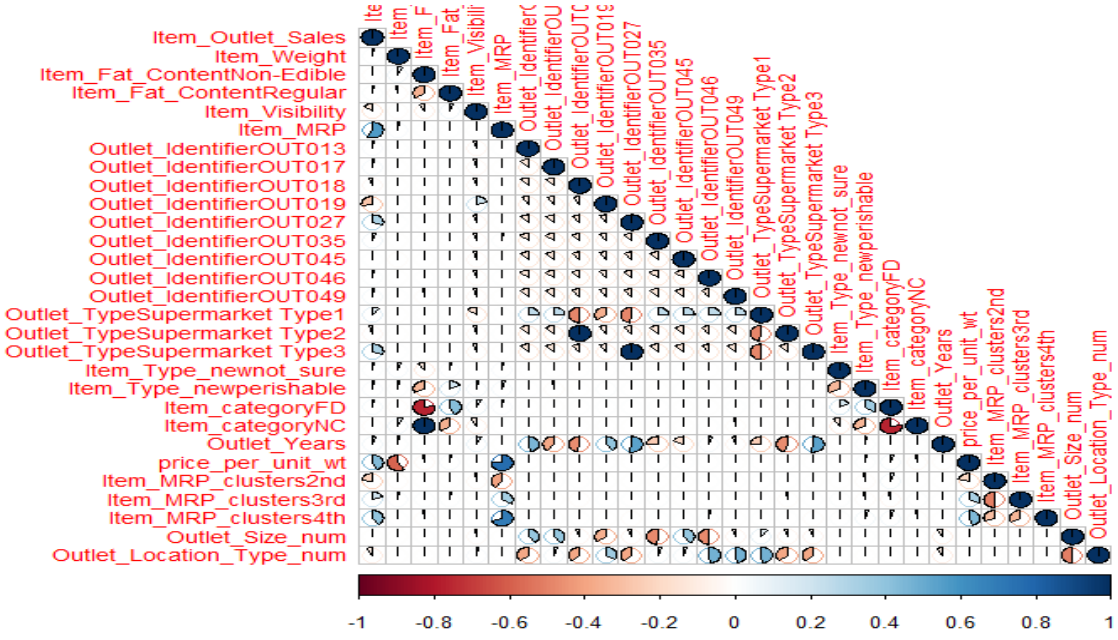
```
num_vars = which(sapply(combi, is.numeric)) # index of numeric features
> num_vars_names = names(num_vars)
> combi_numeric = combi[,setdiff(num_vars_names, "Item_Outlet_Sales"), with = F]
> prep_num = preProcess(combi_numeric, method=c("center", "scale"))
> combi_numeric_norm = predict(prep_num, combi_numeric)
> combi[,setdiff(num_vars_names, "Item_Outlet_Sales") := NULL] # removing numeric independent variables
> combi = cbind(combi, combi_numeric_norm)
```

9.3 Correlation Plot:

Lets examine the correlated features of the train dataset. Correlation varies from -1 to 1

1. Negative correlation : < 0 and => -1
2. Positive correlation : > 0 and <= 1
3. No correlation : 0

It is not desirable to have correlation feature if we are using linear regression



The correlation plot above shows the correlation between all the possible pairs of the variables in out data. The correlation bet any two variables is presented by the pie. A bluiesh pie indicates a positive correlation and reddish pie indicates the negative correlation. The magnitude of the correlation is denoted by the area covered by pie.

Variable prices_per_unit_wt and Item_Weight are highly correlated as the formal one was created from the latter. Similarly price_per_unit_wt and Item_MRP are highly correlated for same reason.

10. Model Building:

After Preprocessing Data , the processed data is used to give accurate results by applying multiple algorithms. A model is a set of algorithms that facilitate the process of finding relation between multiple dataset. An effective model can predict accurate results by finding exact insights of data.

Finally we have arrived at most interesting stage of the whole process – predictive modeling. We will start off with start off with the simpler models and gradually move on the more sophisticated models. We will start with the simple linear models and then move over to more complex model like RandomForest & XGBoost.

We will build the following model in the next sections.

1. Linear Regression
2. Lasso Regression
3. Ridge Regression
4. Random Forest
5. XGBoost

Evaluation Metrics for Regression
The purpose of the model is not complete with the evaluation of models performance. That's why we need an evaluation metric to evaluation our model. Since this is the regression problem, we can evaluate our models using any one of the following evaluation metrics:

- **Mean Absolute Error(MAE) is the mean of the absolute value of the errors.**

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

MAE = mean absolute error
y_i = prediction
x_i = true value
n = total number of data points

- **Mean Square Error (MSE) is the mean of the square error**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

- **Root mean square error (RMSE) is the square root of the mean of squared error.**

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}$$

RMSD = root-mean-square deviation
i = variable i
N = number of non-missing data points
x_i = actual observations time series
$\hat{x}_i$ = estimated time series

10.1 Linear Regression:

Linear regression algorithm tries to predict the results by plotting the graph between an independent variable and a dependent variable that are derived from the dataset. It is a general statistical
analysis mechanism used to build machine learning models. The general equation for linear regression is

$Z = a + bE$

Where, Z is the dependent variable and E is independent variable.

However, as per the correlation plot above, we have a few highly correlated independent variables in our data. This issue of multicollinearity can be dealt with regularization.
For the time being, let's build our linear regression model with all the variables.

Results

RMSE	Rsquare
1127.389	0.5634979

10.2 Regularized Model:

Regularised regression models can handle the correlated independent variables well and helps in overcoming overfitting. **Ridge** penalty shrinks the coefficients of correlated predictors towards each other while the **Lasso** tends to pick one of a pair of correlated features and discard the other. The tuning parameter **lambda** controls the strength of the penalty.

- **Result of Lasso Regression**

RMSE	Rsquare
1127.38	0.5635047

- **Result of Ridge Regression**

RMSE	Rsquare
1132.801	0.5592971

10.3 RandomForest:

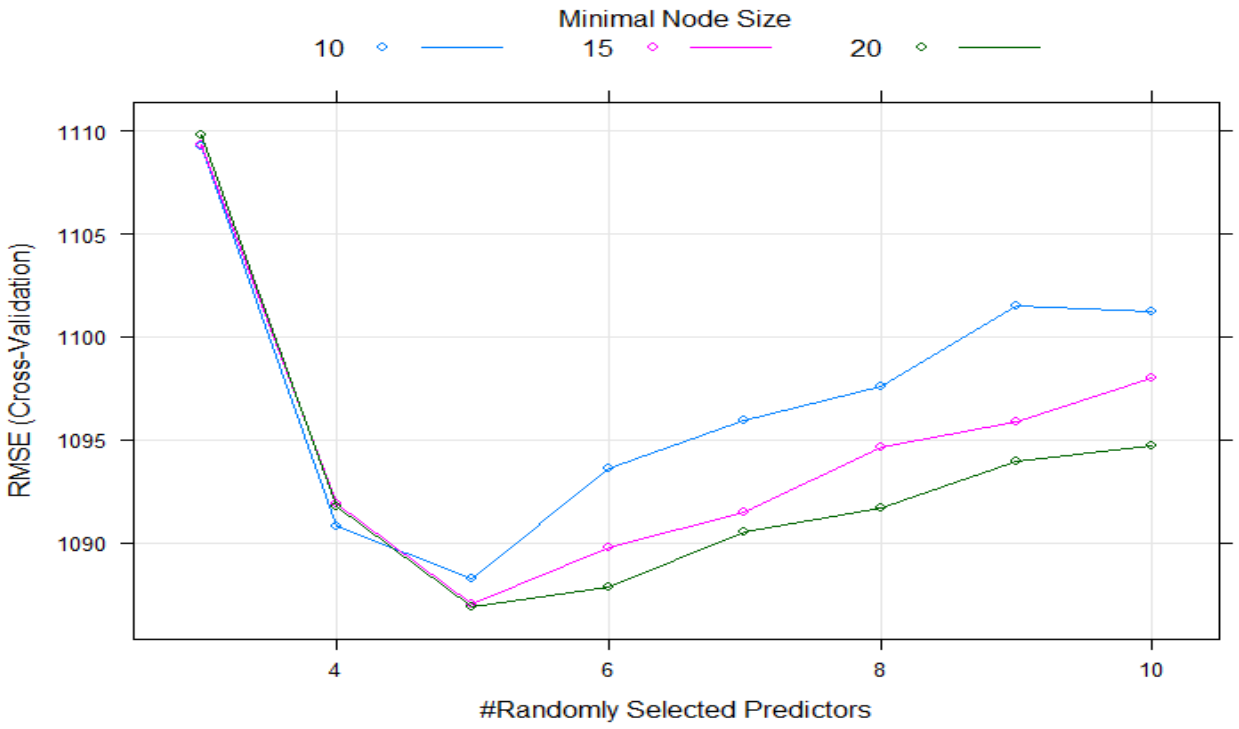
Random Forest Algorithm is used to incorporate predictions from multiple decision trees into a single model. This algorithm uses bagging mechanism to create a forest of decision trees. It the incorporates the predictions from multiple decision trees to give very accurate predictions. The Random
Forest algorithm has two steps involved,

- Random forest formation.
- Predict by Random forest classifier generated.

RandomForest is a tree based bootstrapping algorithm wherein a certain no. of weak learners (decision trees) are combined to make a powerful prediction model. For every indivisual learner, a random sample of rows and a few randomly chosen variables are used to build a decision tree model. Final prediction can be a function of all the predictions made by the individual learners. In case of regression problem, the final prediction can be mean of all the predictions.

We will now build a RandomForest model with 400 trees. The other tuning parameters used here are mtry — no. of predictor variables randomly sampled at each split, and min.node.size — minimum size of terminal nodes (setting this number large causes smaller trees and reduces overfitting).

10.3.1 Best Model Parameters:



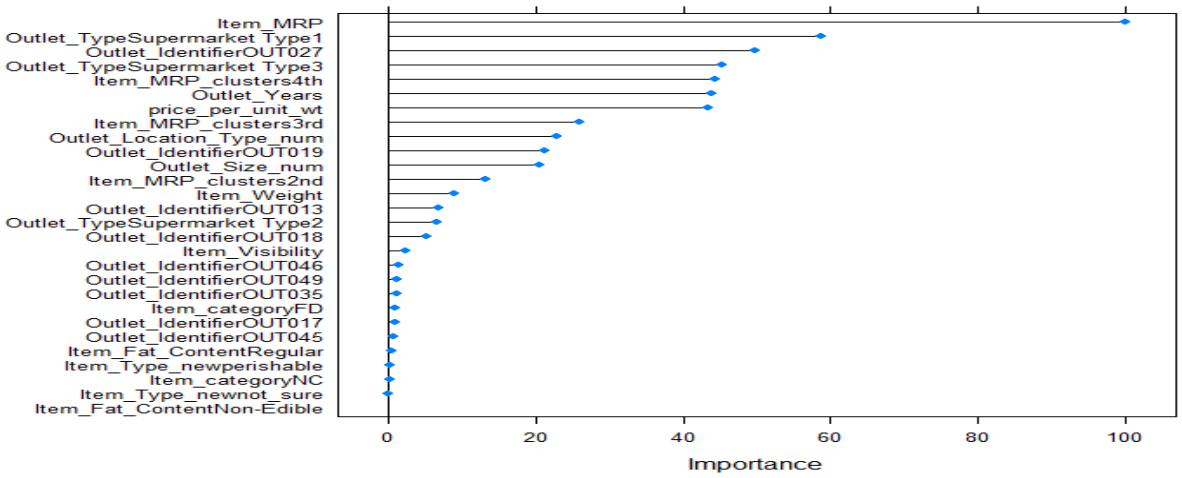
As per the plot shown above, the best score is achieved at mtry = 5 and min.node.size = 20.

Result:

RMSE	Rsquare
917.175	0.711103

10.3.1 Variables Importance:

Lets plot features importance based on the RandomForest model



As expected Item_MRP is the most important variable in predicting the target variable. New features created by us, Like price_per_unit_wt, Outlet_year,Item_MRP_cluster,are also among the top most important variables. This is why feature feature engineering play such a crucial role in predictive modelling.

10.4 XGBoost Model:

XGBoost is a fast and efficient algorithm and has been used to by the winners of many data science competetions. It's a boosting algorithm and you may refer this article to know more about boosting.XGBoost works only with numeric variables and we have already done that. There are many tuning parameters in XGBoost which can be broadly classified into General Parameters, Booster Parameters and Task Parameters.

11 General parameters refers to which booster we are using to do boosting. The commonly used are tree or linear model
12 Booster parameters depends on which booster you have chosen
13 Learning Task parameters that decides on the learning scenario, for example, regression tasks may use different parameters with ranking tasks.

Let's have a look at the parameters that we are going to use in our model.

1. **eta**: It is also known as the learning rate or the shrinkage factor. It actually shrinks the feature weights to make the boosting process more conservative. The range is 0 to 1. Low eta value means model is more robust to overfitting.
2. **gamma**: The range is 0 to ∞. Larger the gamma more conservative the algorithm is.
3. **max_depth**: We can specify maximum depth of a tree using this parameter.
4. **subsample**: It is the proportion of rows that the model will randomly select to grow trees.
5. **colsample_bytree**: It is the ratio of variables randomly chosen for build each tree in the model.

10.4.1 Cross Validation:

We are going to use the xgb() function for cross validation. This function comes with the xgboost package itself.Here we are using cross validation for finding the optimal value of nrounds.

Results

```
 [441]  train-rmse:965.548401+8.568953   test-rmse:1090.328393+27.324710
[451]   train-rmse:963.030750+8.614589   test-rmse:1090.406641+27.158451
Stopping. Best iteration:
[429]   train-rmse:968.550476+8.675415   test-rmse:1090.213550+27.261872
```

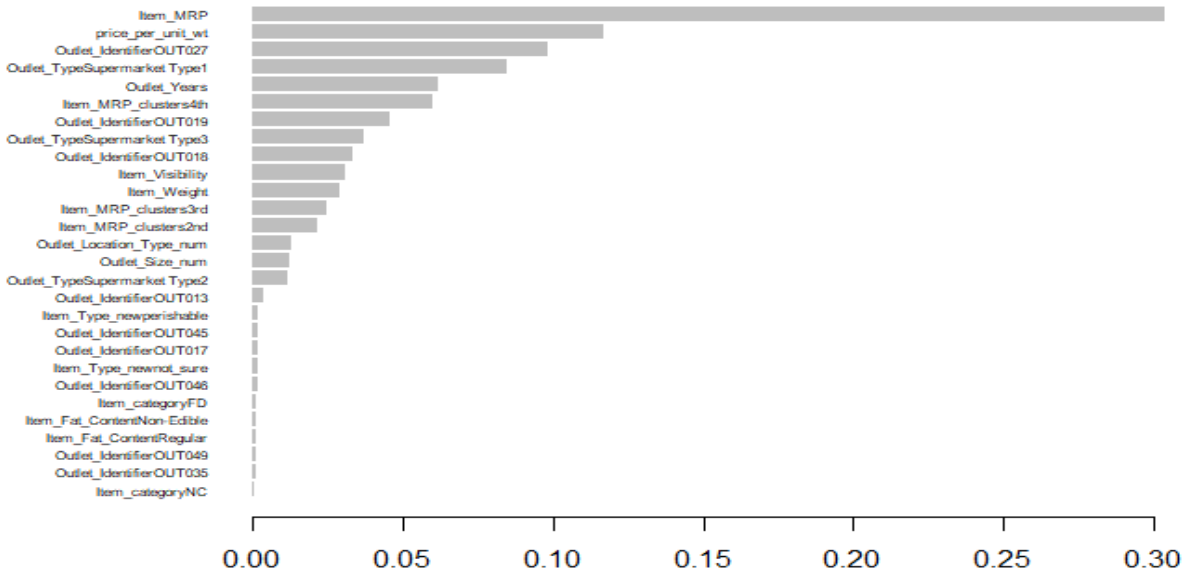
10.4.2 Model Training:

As per the above, we got validation/test score at the 429th iteration. Hence, we will use nround = 429 for building the XGBoost model.

Results : RMSE train – 968.55 test- 1090.21

This model has even outperformed the RandomForest model.

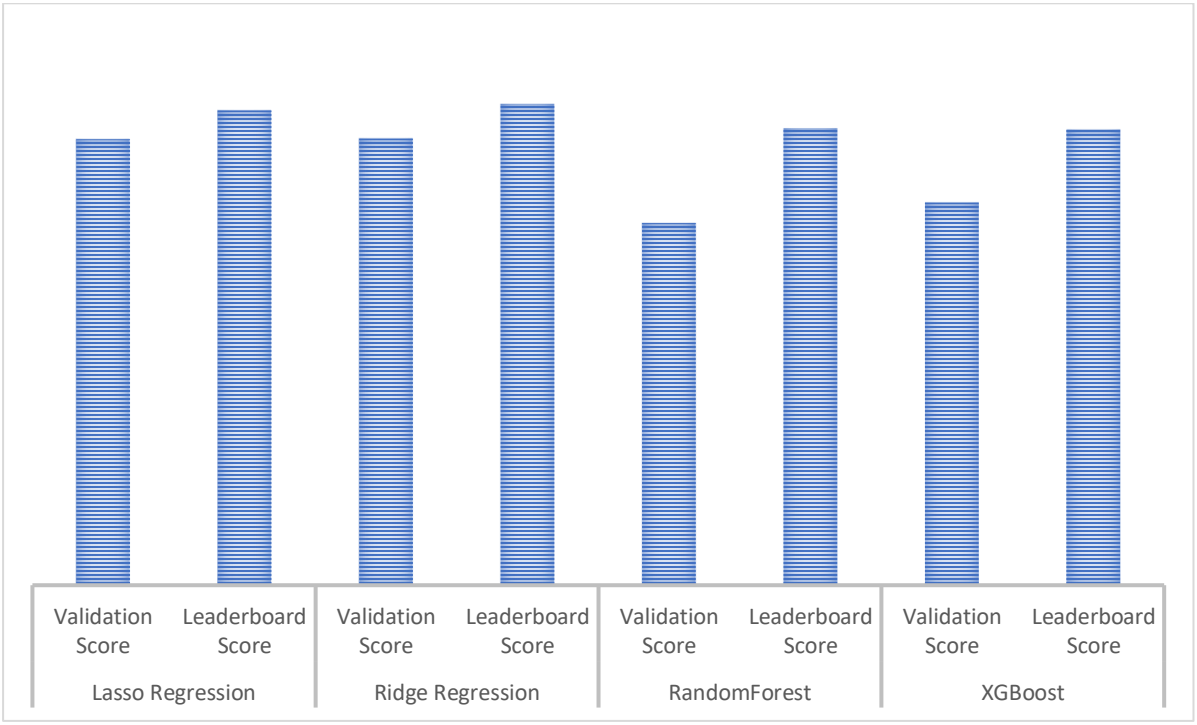
10.4.3 Variables Importance:



As expected Item_MRP is the most important variable in predicting the target variable. New features created by us, Like price_per_unit_wt, Outlet_year,Item_MRP_cluster,are also among the top most important variables.

11.Summary:

After trying and testing 5 different algorithms, the best score on the public leaderboard is achieved by XGBoost (968.55), followed by RandomForest (917.17). However, there are still a plenty of things that we can try to further improve our predictions



12. Conclusion:

The objective of this framework is to predict the future sales from given data of the previous year's using machine Learning techniques. In this paper, we have discussed how different machine learning models are built using different algorithms like Linear regression, Random forest regressor, and XG booster algorithms. These algorithms have been applied to predict the final result of sales. We have addressed in detail about how the noisy data is been removed and the algorithms used to predict the result. Based on the accuracy predicted by different models we conclude that the random forest approach and XG Booster approach are best models. Our predictions help big marts to refine their methodologies and strategies which in turn helps them to increase their profit.

13. Appendix

Github link:

https://github.com/vicky61992/Statistics_learning_Supervised_Project

```
# Load required libraries

library("leaps")
library("ggplot2")
library("reshape2")
library("MASS")
library("ggcorrplot")
library("plotmo")
library("dplyr")
library("gridExtra")
library("Simpsons")
library("GGally")
library("memisc")
library("pander")
library("caret")
library("glmnet")
library("mlbench")
library("psych")
library(data.table) # used for reading and manipulation of data
library(caret)      # used for modeling
library(xgboost)    # used for building XGBoost model
library(e1071)      # used for skewness
library(cowplot)    # used for combining multiple plots
library(corrplot)
```

```
# Data importing and processing

getwd()
setwd("C:/Users/VSBAG/Desktop/DSE_Milan/3rd_sem_subject/ML&SL/SL_Project/My_
Supervised_LEarning/S_SL_Project")
# Data Loading
train <-
fread("https://raw.githubusercontent.com/vicky61992/Statistics_learning_Supervised_
Project/main/Train.csv")
test <-
fread("https://raw.githubusercontent.com/vicky61992/Statistics_learning_Supervised_
Project/main/Test.csv")
submission <-
fread("https://raw.githubusercontent.com/vicky61992/Statistics_learning_Supervised_
Project/main/sample_submission.csv")
#Dimensions of Data
dim(train)
dim(test)

#Structure of Data
str(train)
str(test)

#Features of Data
head(train)
head(test)
names(train)
names(test)


# summary
summary(train)
summary(test)

#Null value

sapply(train,function(x)sum(is.na(x)))
sapply(test,function(x)sum(is.na(x)))

#Combine Train and Test (To save TIME)

test[,Item_Outlet_Sales := NA]
combi = rbind(train, test) # combining train and test datasets
dim(combi)


#Exploratory Data Analysis(EDA)
```

```
ggplot(train) + geom_histogram(aes(train$Item_Outlet_Sales), binwidth = 100, fill =
"pink") +
  xlab("Item_Outlet_Sales")


#Independent Variables(numeric variables) on Train data

p1 = ggplot(combi) + geom_histogram(aes(Item_Weight), binwidth = 0.5, fill = "blue")
p2 = ggplot(combi) + geom_histogram(aes(Item_Visibility), binwidth = 0.005, fill =
"blue")
p3 = ggplot(combi) + geom_histogram(aes(Item_MRP), binwidth = 1, fill = "blue")
plot_grid(p1, p2, p3, nrow = 1) # plot_grid() from cowplot package


#Notes
#1There seems to be no clear-cut pattern in Item_Weight.
#2Item_Visibility is right-skewed and should be transformed to curb its skewness.
#3We can clearly see 4 different distributions for Item_MRP. It is an interesting insight.

#Independent Variables(categorical variables)

ggplot(combi%>% group_by(Item_Fat_Content) %>% summarise(Count = n())) +
  geom_bar(aes(Item_Fat_Content, Count), stat = "identity", fill = "coral1")


#Cobination of Low fat & Reg

combi$Item_Fat_Content[combi$Item_Fat_Content == "LF"] = "Low Fat"
combi$Item_Fat_Content[combi$Item_Fat_Content == "low fat"] = "Low Fat"
combi$Item_Fat_Content[combi$Item_Fat_Content == "reg"] = "Regular"

#Recrate Independent Variables(categorical variables)
ggplot(combi %>% group_by(Item_Fat_Content) %>% summarise(Count = n())) +
  geom_bar(aes(Item_Fat_Content, Count), stat = "identity", fill = "coral1")

## check the other categorical variables

# plot for Item_Type
p4 = ggplot(combi %>% group_by(Item_Type) %>% summarise(Count = n())) +
  geom_bar(aes(Item_Type, Count), stat = "identity", fill = "coral1") +
  xlab("") +
  geom_label(aes(Item_Type, Count, label = Count), vjust = 0.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  ggtitle("Item_Type")

# plot for Outlet_Identifier
p5 = ggplot(combi %>% group_by(Outlet_Identifier) %>% summarise(Count = n())) +
  geom_bar(aes(Outlet_Identifier, Count), stat = "identity", fill = "coral1") +
  geom_label(aes(Outlet_Identifier, Count, label = Count), vjust = 0.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# plot for Outlet_Size
```

```
p6 = ggplot(combi %>% group_by(Outlet_Size) %>% summarise(Count = n())) +
  geom_bar(aes(Outlet_Size, Count), stat = "identity", fill = "coral1") +
  geom_label(aes(Outlet_Size, Count, label = Count), vjust = 0.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

second_row = plot_grid(p5, p6, nrow = 1)
plot_grid(p4, second_row, ncol = 1)

#remaining categorical variables
# plot for Outlet_Establishment_Year
p7 = ggplot(combi %>% group_by(Outlet_Establishment_Year) %>% summarise(Count
= n())) +
  geom_bar(aes(factor(Outlet_Establishment_Year), Count), stat = "identity", fill =
"coral1") +
  geom_label(aes(factor(Outlet_Establishment_Year), Count, label = Count), vjust = 0.5)
+
  xlab("Outlet_Establishment_Year") +
  theme(axis.text.x = element_text(size = 8.5))
# plot for Outlet_Type
p8 = ggplot(combi %>% group_by(Outlet_Type) %>% summarise(Count = n())) +
  geom_bar(aes(Outlet_Type, Count), stat = "identity", fill = "coral1") +
  geom_label(aes(factor(Outlet_Type), Count, label = Count), vjust = 0.5) +
  theme(axis.text.x = element_text(size = 8.5))

# ploting both plots together
plot_grid(p7, p8, ncol = 2)

#Bivariate Analysis
#we'll explore the independent variables with respect to the target variable

train = combi[1:nrow(train)] # extracting train data from the combined data

#Target Variable vs Independent Numerical Variables
# Item_Weight vs Item_Outlet_Sales
p9 = ggplot(train) +
  geom_point(aes(Item_Weight, Item_Outlet_Sales), colour = "violet", alpha = 0.3) +
  theme(axis.title = element_text(size = 8.5))

# Item_Visibility vs Item_Outlet_Sales
p10 = ggplot(train) +
  geom_point(aes(Item_Visibility, Item_Outlet_Sales), colour = "violet", alpha = 0.3) +
  theme(axis.title = element_text(size = 8.5))

# Item_MRP vs Item_Outlet_Sales
p11 = ggplot(train) +
  geom_point(aes(Item_MRP, Item_Outlet_Sales), colour = "violet", alpha = 0.3) +
  theme(axis.title = element_text(size = 8.5))
second_row_2 = plot_grid(p10, p11, ncol = 2)
plot_grid(p9, second_row_2, nrow = 2)

#Target Variable vs Independent Categorical Variables
```

```
# Item_Type vs Item_Outlet_Sales
p12 = ggplot(train) +
  geom_violin(aes(Item_Type, Item_Outlet_Sales), fill = "magenta") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text = element_text(size = 6),
        axis.title = element_text(size = 8.5))

# Item_Fat_Content vs Item_Outlet_Sales
p13 = ggplot(train) +
  geom_violin(aes(Item_Fat_Content, Item_Outlet_Sales), fill = "magenta") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text = element_text(size = 8),
        axis.title = element_text(size = 8.5))

# Outlet_Identifier vs Item_Outlet_Sales
p14 = ggplot(train) +
  geom_violin(aes(Outlet_Identifier, Item_Outlet_Sales), fill = "magenta") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text = element_text(size = 8),
        axis.title = element_text(size = 8.5))
second_row_3 = plot_grid(p13, p14, ncol = 2)
plot_grid(p12, second_row_3, ncol = 1)

#remaining variables
ggplot(train)+ geom_violin(aes(Outlet_Size, Item_Outlet_Sales),fill = "magenta")
p15 = ggplot(train) + geom_violin(aes(Outlet_Location_Type, Item_Outlet_Sales), fill =
"magenta")
p16 = ggplot(train) + geom_violin(aes(Outlet_Type, Item_Outlet_Sales), fill =
"magenta")
plot_grid(p15, p16, ncol = 1)


#Missing Value Treatment

#There are different methods to treat missing values based on the problem and the
data. Some of the common techniques are as follows:

#1.Deletion of rows: In train dataset, observations having missing values in any
variable are deleted. The downside of this method is the loss of information and drop
in prediction power of model.

#2.Mean/Median/Mode Imputation: In case of continuous variable, missing values can
be replaced with mean or median of all known values of that variable. For categorical
variables, we can use mode of the given values to replace the missing values.

#3.Building Prediction Model: We can even make a predictive model to impute missing
data in a variable. Here we will treat the variable having missing data as the target
variable and the other variables as predictors. We will divide our data into 2 datasets-
one without any missing value for that variable and the other with missing values for
that variable. The former set would be used as training set to build the predictive
model and it would then be applied to the latter set to predict the missing values.
```

```
#find missing values in a variable.

sum(is.na(combi$Item_Weight))

#Imputing Missing Value

missing_index = which(is.na(combi$Item_Weight))
for(i in missing_index){

  item = combi$Item_Identifier[i]
  combi$Item_Weight[i] = mean(combi$Item_Weight[combi$Item_Identifier == item],
na.rm = T)
}

#Cross Check

sum(is.na(combi$Item_Weight))

#Replacing 0's in Item_Visibility variable

ggplot(combi) + geom_histogram(aes(Item_Visibility), bins = 100)

#replace the zeroes

zero_index = which(combi$Item_Visibility == 0)
for(i in zero_index){

  item = combi$Item_Identifier[i]
  combi$Item_Visibility[i] = mean(combi$Item_Visibility[combi$Item_Identifier == item],
na.rm = T)

}

ggplot(combi) + geom_histogram(aes(Item_Visibility), bins = 100)


sapply(combi,function(x)sum(is.na(x)))


#Feature Engineering

#will create the following new features:

#1Item_Type_new: Broader categories for the variable Item_Type.
#2Item_category: Categorical variable derived from Item_Identifier.
#3Outlet_Years: Years of operation for outlets.
#4price_per_unit_wt: Item_MRP/Item_Weight
#5Item_MRP_clusters: Binned feature for Item_MRP.

#Item_Type variable and classify the categories into perishable and non_perishable as
per our understanding and make it into a new feature.
```

```
perishable = c("Breads", "Breakfast", "Dairy", "Fruits and Vegetables", "Meat",
"Seafood")
non_perishable = c("Baking Goods", "Canned", "Frozen Foods", "Hard Drinks",
"Health and Hygiene", "Household", "Soft Drinks")

# create a new feature 'Item_Type_new'
combi[,Item_Type_new := ifelse(Item_Type %in% perishable, "perishable",
ifelse(Item_Type %in% non_perishable, "non_perishable", "not_sure"))]

#Let's compare Item_Type with the first 2 characters of Item_Identifier, i.e., 'DR', 'FD',
and 'NC'. These identifiers most probably stand for drinks, food, and non-consumable.

table(combi$Item_Type, substr(combi$Item_Identifier, 1, 2))

# Item_category Created

combi[,Item_category := substr(combi$Item_Identifier, 1, 2)]

#Outlet_Years (years of operation) and price_per_unit_wt (price per unit weight).

combi$Item_Fat_Content[combi$Item_category == "NC"] = "Non-Edible"

# years of operation for outlets
combi[,Outlet_Years := 2013 - Outlet_Establishment_Year]
combi$Outlet_Establishment_Year = as.factor(combi$Outlet_Establishment_Year)

# Price per unit weight
combi[,price_per_unit_wt := Item_MRP/Item_Weight]

#the Item_MRP vs Item_Outlet_Sales plot, we saw Item_MRP was spread across in 4
chunks. Now let's assign a label to each of these chunks and use this label as a new
variable.

# creating new independent variable - Item_MRP_clusters
combi[,Item_MRP_clusters := ifelse(Item_MRP < 69, "1st",
                           ifelse(Item_MRP >= 69 & Item_MRP < 136, "2nd",
                                ifelse(Item_MRP >= 136 & Item_MRP < 203, "3rd", "4th")))]

#Encoding Categorical Variables

#Most of the machine learning algorithms produce better result with numerical
variables only. So, it is essential to treat the categorical variables present in the data

#We will use 2 techniques - Label Encoding and One Hot Encoding.

#1Label encoding simply means converting each category in a variable to a number. It
is more suitable for ordinal variables - categorical variables with some order.

#2In One hot encoding, each category of a categorical variable is converted into a new
binary column (1/0).

#Label encoding for the categorical variables
```

```
str(combi)

combi[,Outlet_Size_num := ifelse(Outlet_Size == "Small", 0,
                                 ifelse(Outlet_Size == "Medium", 1,2))]
combi[,Outlet_Location_Type_num := ifelse(Outlet_Location_Type == "Tier 3", 0,
                                          ifelse(Outlet_Location_Type == "Tier 2", 1, 2))]
# removing categorical variables after label encoding
combi[, c("Outlet_Size", "Outlet_Location_Type") := NULL]

#One hot encoding for the categorical variable

ohe = dummyVars("~.", data = combi[,-c("Item_Identifier",
"Outlet_Establishment_Year", "Item_Type")], fullRank = T)
ohe_df = data.table(predict(ohe, combi[,-c("Item_Identifier",
"Outlet_Establishment_Year", "Item_Type")]))
combi = cbind(combi[,"Item_Identifier"], ohe_df)



#PreProcessing Data


skewness(combi$Item_Visibility); skewness(combi$price_per_unit_wt)


#Removing Skewness

combi[,Item_Visibility := log(Item_Visibility + 1)] # log + 1 to avoid division by zero
combi[,price_per_unit_wt := log(price_per_unit_wt + 1)]



#Scaling numeric predictors

#Let's scale and center the numeric variables to make them have a mean of zero,
standard deviation of one and scale of 0 to 1. Scaling and centering is required for
linear regression models.

num_vars = which(sapply(combi, is.numeric)) # index of numeric features
num_vars_names = names(num_vars)
combi_numeric = combi[,setdiff(num_vars_names, "Item_Outlet_Sales"), with = F]
prep_num = preProcess(combi_numeric, method=c("center", "scale"))
combi_numeric_norm = predict(prep_num, combi_numeric)

combi[,setdiff(num_vars_names, "Item_Outlet_Sales") := NULL] # removing numeric
independent variables
combi = cbind(combi, combi_numeric_norm)



#Splitting the combined data combi back to train and test set.
```

```
train = combi[1:nrow(train)]
test = combi[(nrow(train) + 1):nrow(combi)]
test[,Item_Outlet_Sales := NULL] # removing Item_Outlet_Sales as it contains only NA

# Correlation Plot

cor_train = cor(train[,-c("Item_Identifier")])
corrplot(cor_train, method = "pie", type = "lower", tl.cex = 0.9)

sapply(train,function(x)sum(is.na(x)))
#Model Building

#Linear Regression

#Building Model

linear_reg_mod1 = lm(Item_Outlet_Sales ~ ., data = train[,-c("Item_Identifier")])

summary(linear_reg_mod1)

predtest<- predict(linear_reg_mod1,test[,-c("Item_Identifier")])


#Making Predictions on test Data

head(predtest)
predtest1<-data.frame(predtest)
final_data<- cbind(test,predtest1)
write.csv(final_data,"predicted.csv")


# preparing dataframe for submission and writing it in a csv file

set.seed(1234)
my_control = trainControl(method="cv", number=5)
linear_reg_mod = train(x = train[,-c("Item_Identifier", "Item_Outlet_Sales")], y =
train$Item_Outlet_Sales,
                       method='glmnet', trControl= my_control)
print("5- fold cross validation scores:")
summary(linear_reg_mod)
print(round(linear_reg_mod$resample$RMSE, 2))
submission$Item_Outlet_Sales = predict(linear_reg_mod, test[,-c("Item_Identifier")])
write.csv(submission, "Linear_Reg_submit_2_21_Apr_18.csv", row.names = F)



#Lasso Regression

set.seed(1235)
my_control = trainControl(method="cv", number=5)
Grid = expand.grid(alpha = 1, lambda = seq(0.001,0.1,by = 0.0002))
```

```
lasso_linear_reg_mod = train(x = train[, -c("Item_Identifier", "Item_Outlet_Sales")], y =
train$Item_Outlet_Sales,
                        method='glmnet', trControl= my_control, tuneGrid = Grid)


# mean validation score
mean(lasso_linear_reg_mod$resample$RMSE)

#Ridge Regression

set.seed(1236)
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(x = train[, -c("Item_Identifier", "Item_Outlet_Sales")], y =
train$Item_Outlet_Sales,
                        method='glmnet', trControl= my_control, tuneGrid = Grid)


# mean validation score
mean(ridge_linear_reg_mod$resample$RMSE)


#RandomForest

set.seed(1237)
my_control = trainControl(method="cv", number=5)

tgrid = expand.grid(
  .mtry = c(3:10),
  .splitrule = "variance",
  .min.node.size = c(10,15,20)
)

rf_mod = train(x = train[, -c("Item_Identifier", "Item_Outlet_Sales")],
               y = train$Item_Outlet_Sales,
               method='ranger',
               trControl= my_control,
               tuneGrid = tgrid,
               num.trees = 400,
               importance = "permutation")

# mean validation score
mean(rf_mod$resample$RMSE)

# best model parameter
plot(rf_mod)


# Important variable

plot(varImp(rf_mod))
```

```
#XGBoost

param_list = list(

  objective = "reg:linear",
  eta=0.01,
  gamma = 1,
  max_depth=6,
  subsample=0.8,
  colsample_bytree=0.5
)
dtrain = xgb.DMatrix(data = as.matrix(train[,-c("Item_Identifier", "Item_Outlet_Sales")]),
label= train$Item_Outlet_Sales)
dtest = xgb.DMatrix(data = as.matrix(test[,-c("Item_Identifier")]))

#Cross Validation

set.seed(112)
xgbcv = xgb.cv(params = param_list,
               data = dtrain,
               nrounds = 1000,
               nfold = 5,
               print_every_n = 10,
               early_stopping_rounds = 30,
               maximize = F)

#Model Training

xgb_model = xgb.train(data = dtrain, params = param_list, nrounds = 429)

#Variable Importance

var_imp = xgb.importance(feature_names = setdiff(names(train), c("Item_Identifier",
"Item_Outlet_Sales")),
                         model = xgb_model)
xgb.plot.importance(var_imp)


# Compare all model

knitr::kable(cbind(lm = unlist(tr.lm.eval),
                   rr = unlist(tr.rr.interract.step.eval),
                   quadratic = unlist(tr.qm.eval),
                   rt = unlist(tr.rpart.eval),
                   it = unlist(tr.lm.interract.step.eval),
                   svm = unlist(tr.svm.eval),
                   svm.cv = unlist(tr.svmRadial.eval)) %>% round(3),
             caption = "ALL MODEL COMPARION TABLE")
```