Predicting Sales and its dependency on certain features

 # We are going to install and load several packages

#for reading data, manipulation of data, visualization of data, and finally for modeling.

install.packages(('data.tables'))

library(data.table) # used for reading and manipulation of data

library(dplyr)   # used for data manipulation and joining

library(ggplot2)   # used for ploting

library(caret)   # used for modeling

library(corrplot)   # used for making correlation plot

library(xgboost)   # used for building XGBoost model. XGBoost is a powerful tool to make classification and regression . It used to make predictions and evaluate the credibility of the predictions.

library(cowplot)   # used for combining multiple plots

#We will use read() function of data.table package to read the datasets.Already separated training and testing dataset. A final dataset to apple our model to.

train = fread("Train_UWu5bXk.csv")

test = fread("Test_u94Q5KV.csv")

final = fread("SampleSubmission_TmnO39y.csv")

#Exploring our dataset

dim(train)

#> dim(train)

#[1] 8523   12

dim(test)

#[1] 5681   11

names(train)

[1] "Item_Identifier"   "Item_Weight"   "Item_Fat_Content"   "Item_Visibility"

[5] "Item_Type"   "Item_MRP"   "Outlet_Identifier"   "Outlet_Establishment_Year"

[9] "Outlet_Size"   "Outlet_Location_Type"   "Outlet_Type"   "Item_Outlet_Sales"

names(test)

[1] "Item_Identifier"   "Item_Weight"   "Item_Fat_Content"   "Item_Visibility"

[5] "Item_Type"   "Item_MRP"   "Outlet_Identifier"   "Outlet_Establishment_Year"

[9] "Outlet_Size"   "Outlet_Location_Type"   "Outlet_Type"

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

 # Find the structure

str(train)

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 11 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>

#Combining training and test dataset.

 # Combining train and test sets saves a lot of time and effort because

#if we have to make any modification in the data, we would make the change only in the combined data and not in train and test data separately.

#Later we can always split the combined data back to train and test.

 # It is still a topic for discussion whether the train data and test data should be combined or be kept separated and modifications or changes be applied separately.

 # For this case , lets combine the dataset.

```
test[,Item_Outlet_Sales := NA]

combi = rbind(train, test) # combining train and test datasets

dim(combi)
```

#Lets do exploratory data analysis on our data. It helps us in understanding the nature of data in terms of distribution of the individual variables,

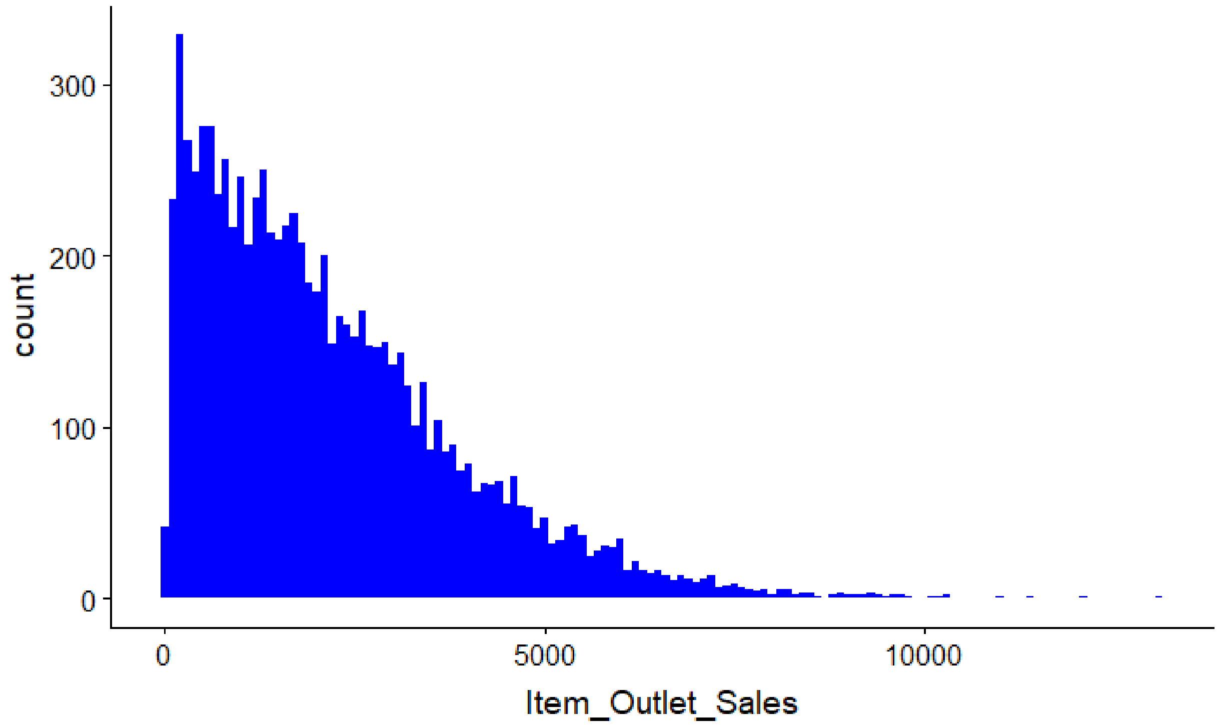
#finding missing values, relationship with other variables etc.

#Let's start with univariate EDA. It involves exploring variables individually.

#We will try to visualize the continuous variables using histograms and categorical variables using bar plots.

#Since our target variable is continuous(item_outlet_Sales), we can visualise it by plotting its histogram.

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

  xlab("Item_Outlet_Sales")
```



As we can see, it is right skewed.

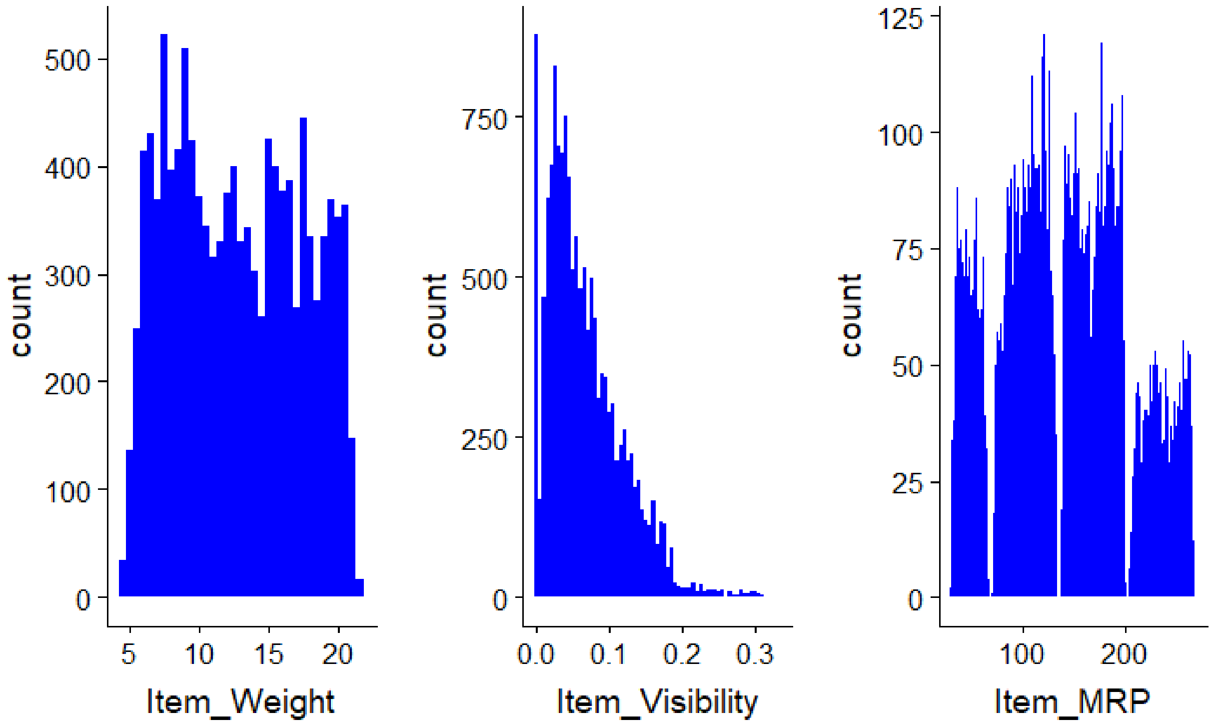
#let's check the numeric independent variables.

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

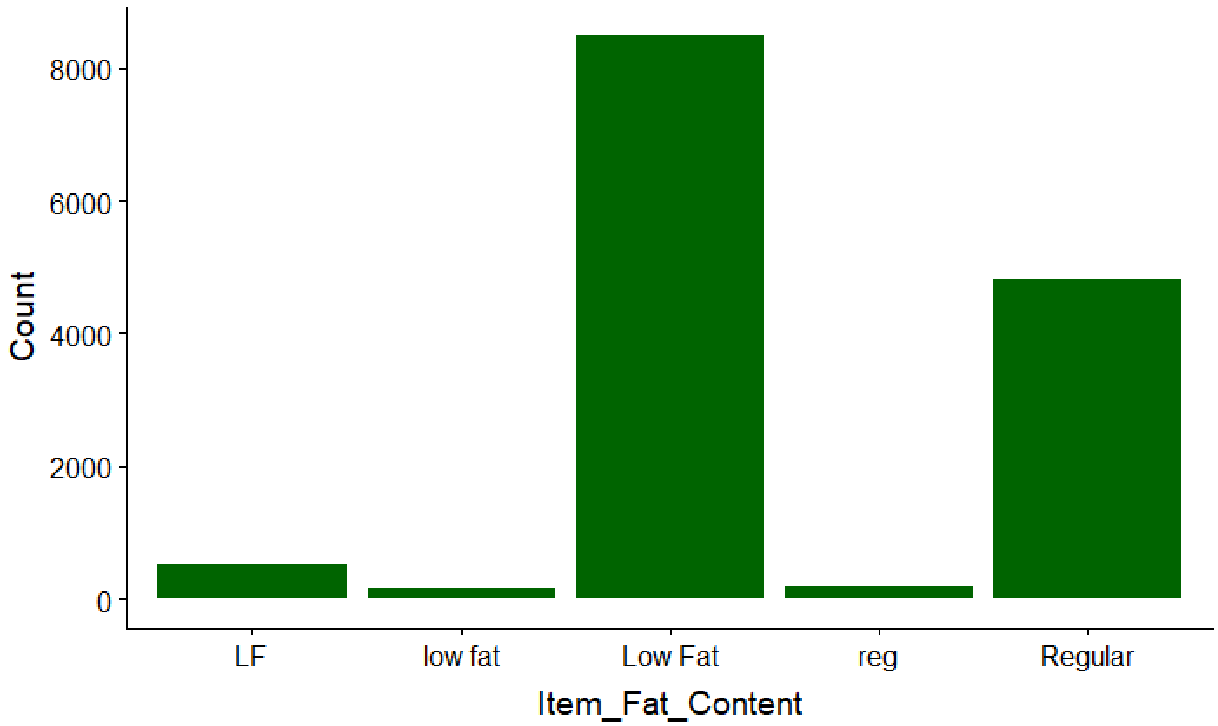


Observations

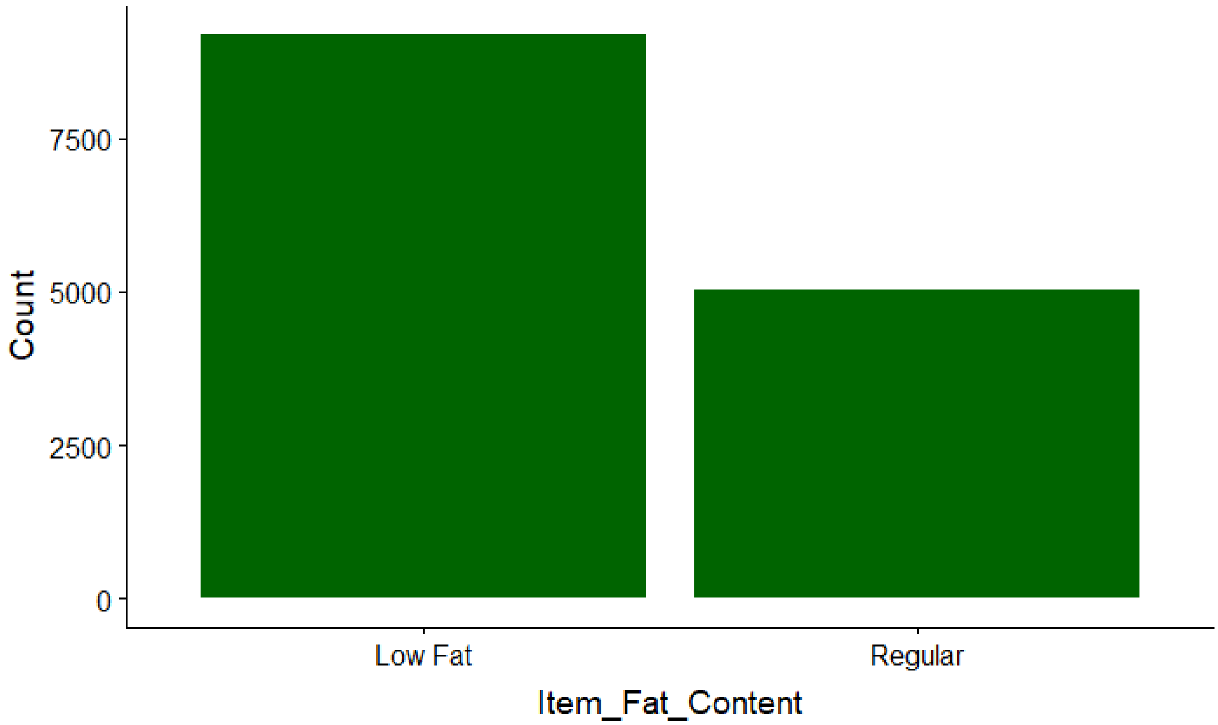
- *There seems to be no clear pattern in Item_Weight.*
- *Item_Visibility is right-skewed.*
- *We can clearly see 4 different distributions for Item_MRP.*

Let's try to explore and gain some insights from the categorical variables.Let's first plot Item_Fat_Content.

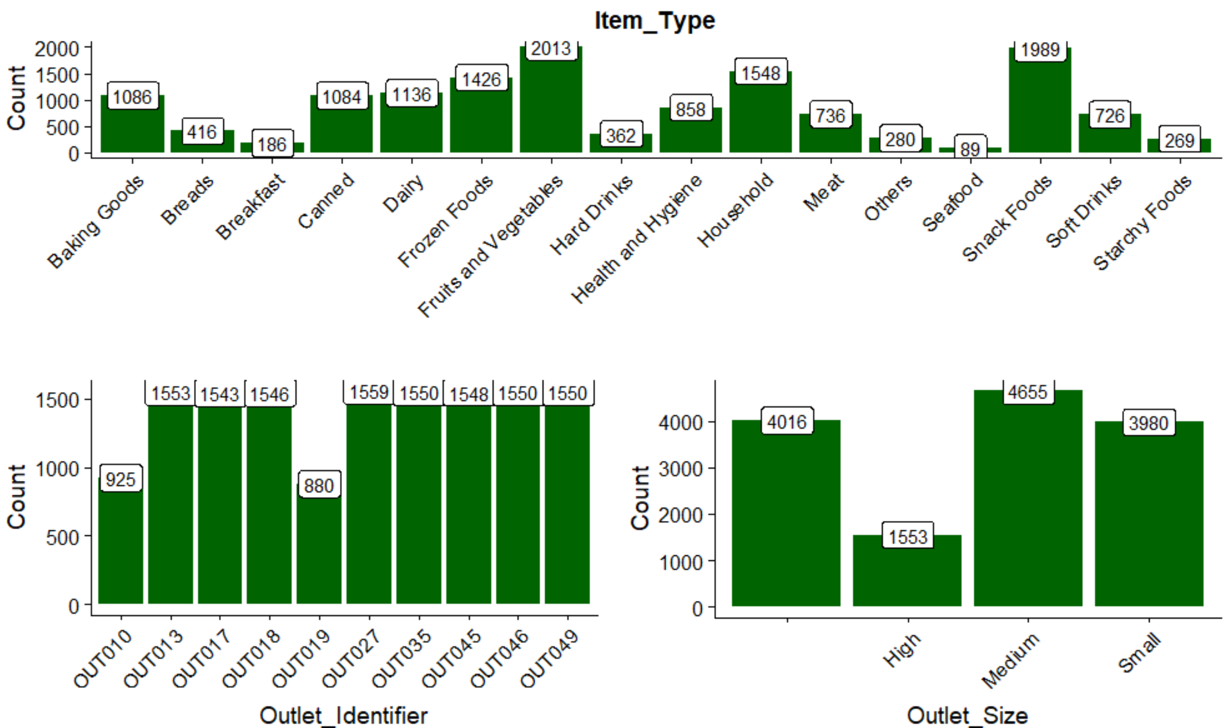
```
ggplot(combi %>% group_by(Item_Fat_Content) %>% summarise(Count = n())) +
geom_bar(aes(Item_Fat_Content, Count), stat = "identity", fill = "darkgreen")
```



In the figure above, 'LF', 'low fat', and 'Low Fat' are the same category and can be combined into one. It can be done for 'reg' and 'Regular' into one. After making these corrections we'll plot the same figure again.




```
# plot for Outlet_Identifier

p5 = ggplot(combi %>% group_by(Outlet_Identifier) %>% summarise(Count = n()))
+  geom_bar(aes(Outlet_Identifier, Count), stat = "identity", fill = "darkgreen") +
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 = "darkgreen") +
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)
```



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

```
#Let's check remaining 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 = "darkgreen")
+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 = "darkgreen") +
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)
```

Observations

- *Lesser number of outlets established in the year 1998 and more in 1985 as compared to the other years.*
- *Supermarket Type 1 has the highest number in Outlet_Type.*

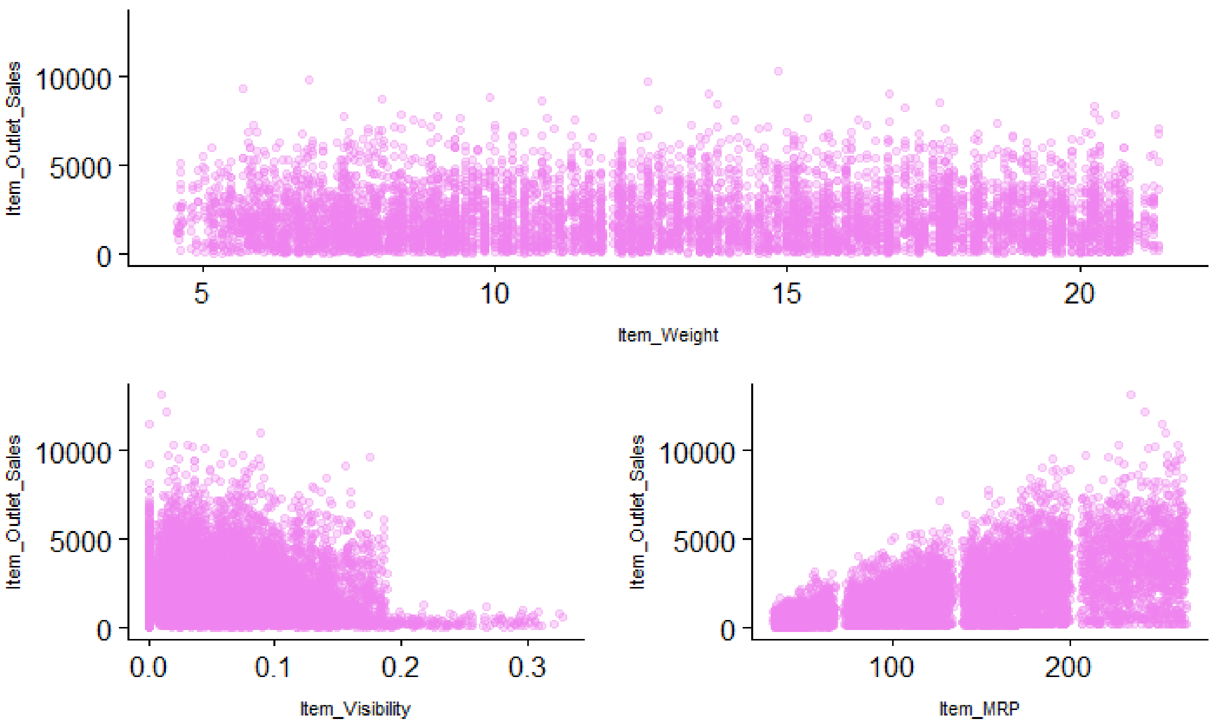
After looking at every feature individually, let's now do some bivariate analysis. Here we'll explore the independent variables with respect to the target variable. The objective is to discover hidden relationships between the independent variable and the target variable.

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

Let's explore the numerical variables first.

```
# 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)
```



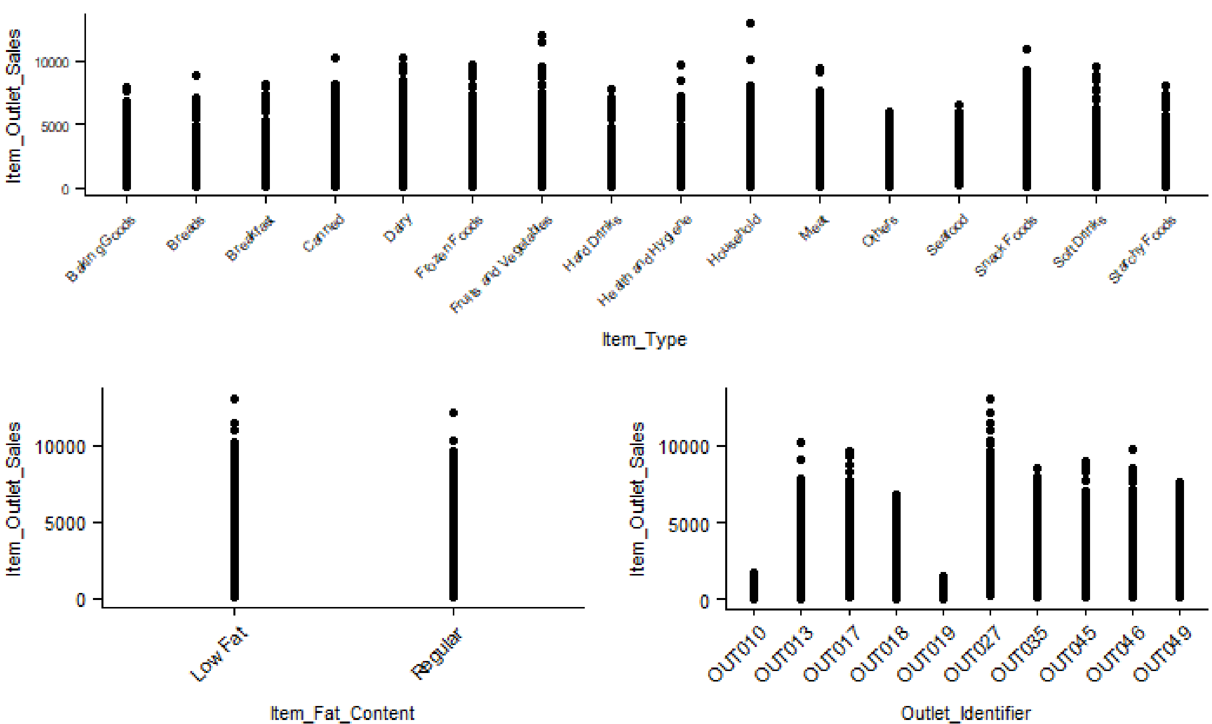
Observations

- *Item_Outlet_Sales is spread across entire range of the Item_Weight without any pattern.*
- *In Item_Visibility vs Item_Outlet_Sales, there is a string of points at Item_Visibility = 0.0 which seems strange as item visibility cannot be completely zero.*
- *In the third plot of Item_MRP vs Item_Outlet_Sales, we can clearly see 4 segments of prices.*

Now we'll visualise the categorical variables with respect to Item_Outlet_Sales.

 # Item_Type vs Item_Outlet_Sales

```r
p12 = ggplot(train) + geom_point(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_point(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_point(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)
```



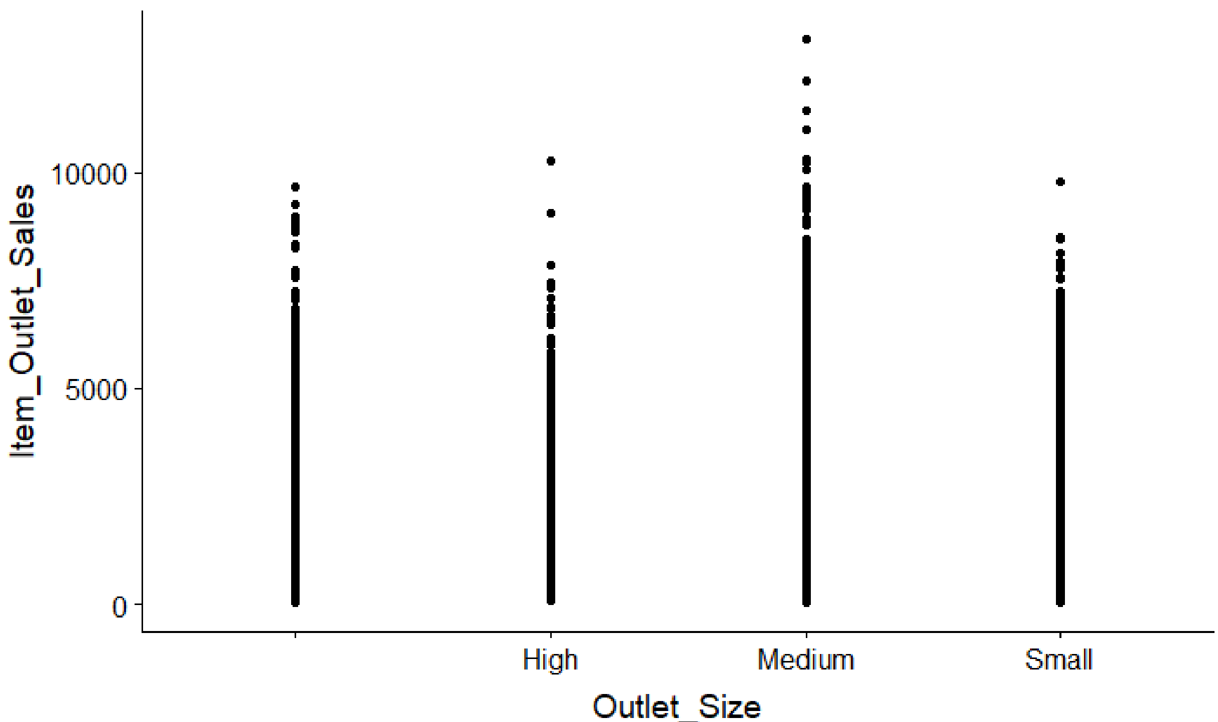
Observations

- *Distribution of Item_Outlet_Sales across the categories of Item_Type is not very distinct and same is the case with Item_Fat_Content.*
- *The distribution for OUT010 and OUT019 categories of Outlet_Identifier are quite similar and very much less from the rest of the categories of Outlet_Identifier.*

Let's check the distribution of the target variable across Outlet_Size.

#Outlet_size vs Item_Outlet_Sale

ggplot(train) + geom_point(aes(Outlet_Size, Item_Outlet_Sales), fill = "magenta")



The distribution of 'Small' Outlet_Size is almost identical to the distribution of the blank category of Outlet_Size. So, we can substitute the blanks in Outlet_Size with 'Small'.

Working on Missing Values

Missing data can have a severe impact on building predictive model because the missing values might contain some vital information which could help in making better predictions. So, it becomes important to work on missing data.

 # Treating Missing Values

sum(is.na(combi$Item_Weight))

#[1] 2439

As you can see above, we have missing values in Item_Weight. We'll now impute Item_Weight *with mean weight based on the* Item_Identifier *variable.*

 # Imputing missing value

```
missing_index = which(is.na(combi$Item_Weight))
for(i in missing_index)
item = combi$Item_Identifier[icombi$Item_Weight[i] =
mean(combi$Item_Weight[combi$Item_Identifier == item], na.rm = T)}
```

Let's check if any null value remains.

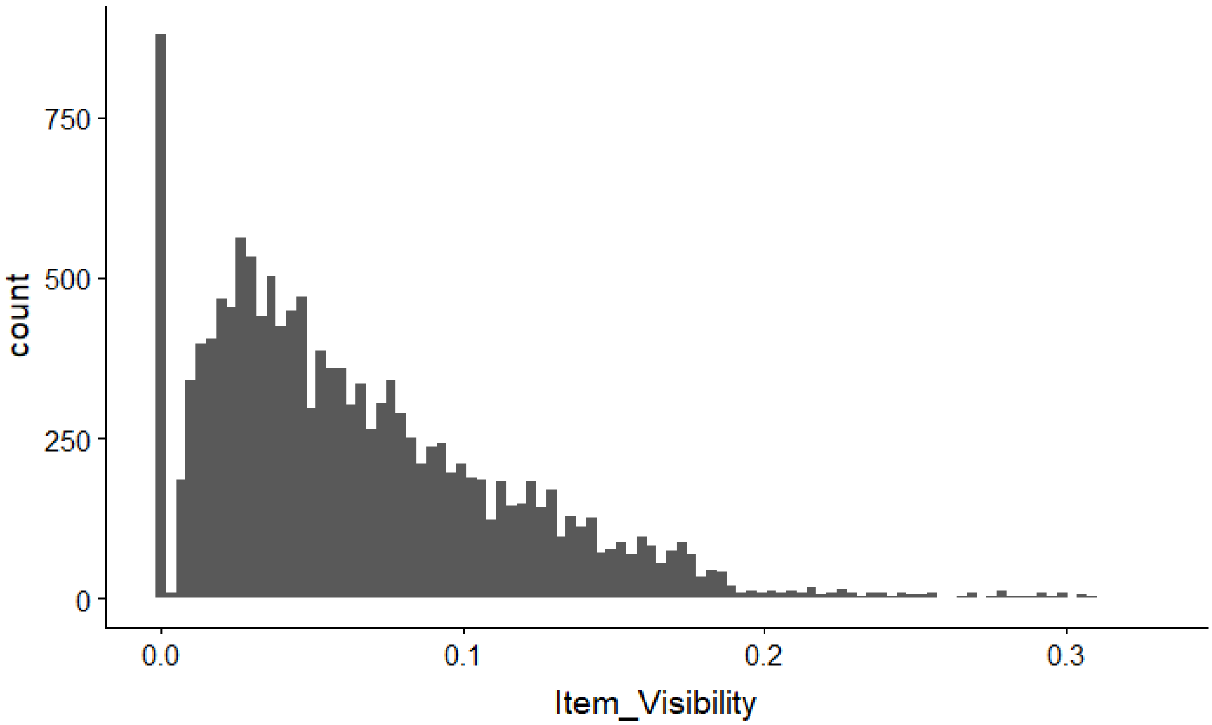
```
sum(is.na(combi$Item_Weight))
#[1] 0
```

0 missing values!

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

Let's first plot the graph for visibility to check missing values

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



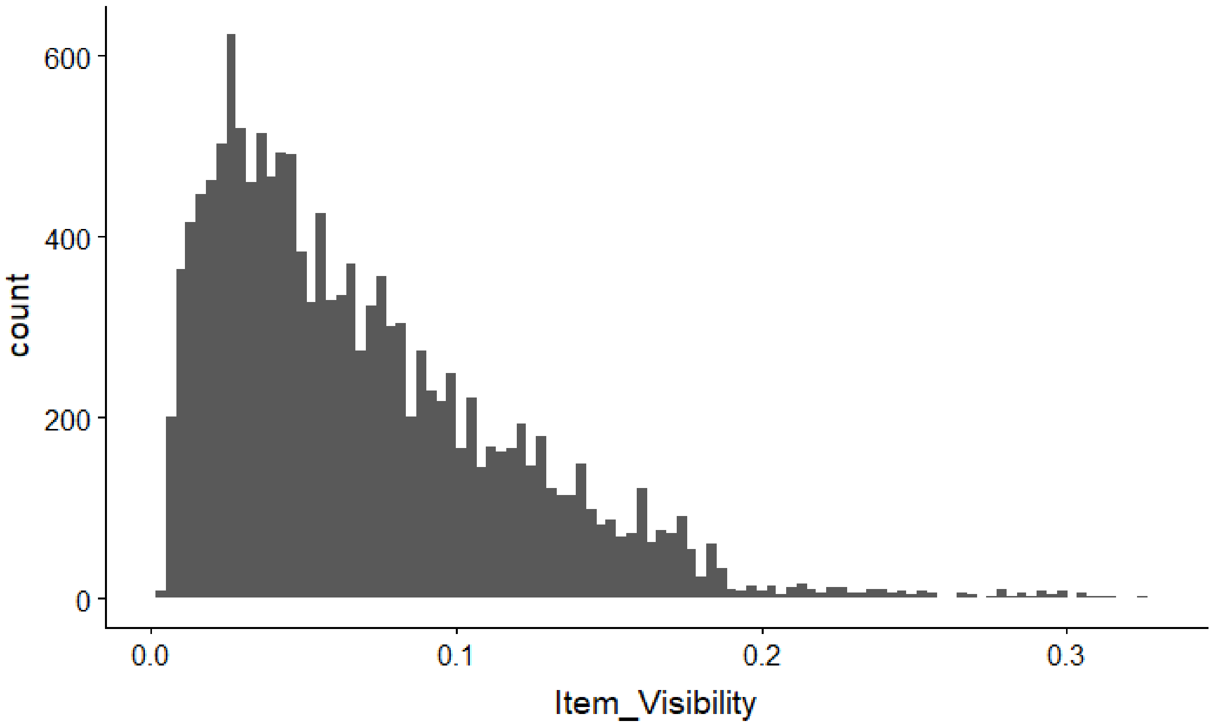
We can clearly see zeroes in this graph.

#replacing zeroes with mean of visibility on basis of item_identifier

```
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)

  }
```

Plotting graph again to check.

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



0 in item visibility are gone!

Converting categorical variables to numerical variables for correlation and regression.

We will use label encoding and one hot encoding to the above

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)*

We will label encode Outlet_Size and Outlet_Location_Type as these are ordinal variables.

#Label Encoding

```
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))]
#One hot Encoding
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)
# removing categorical variables after label encoding
combi[, c("Outlet_Size", "Outlet_Location_Type") := NULL]
```

Preprocessing DATA

Preprocessing is final scaling , cleaning of data before it is fed to the algorithm. If we remember , in our data, variables Item_Visibility is highly skewed. So, we will treat skewness with the help of log transformation.

Log Transformations formula

$\ln Y_i = \beta_1 + \beta_2 \ln X_i + \epsilon_i$

$Y_i = \exp(\beta_1 + \beta_2 \ln X_i) \cdot \exp(\epsilon_i)$

#Let's try remove skweness in item_visibility

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

 # Splitting the datasets back to train and test

train = combi[1:nrow(train)]

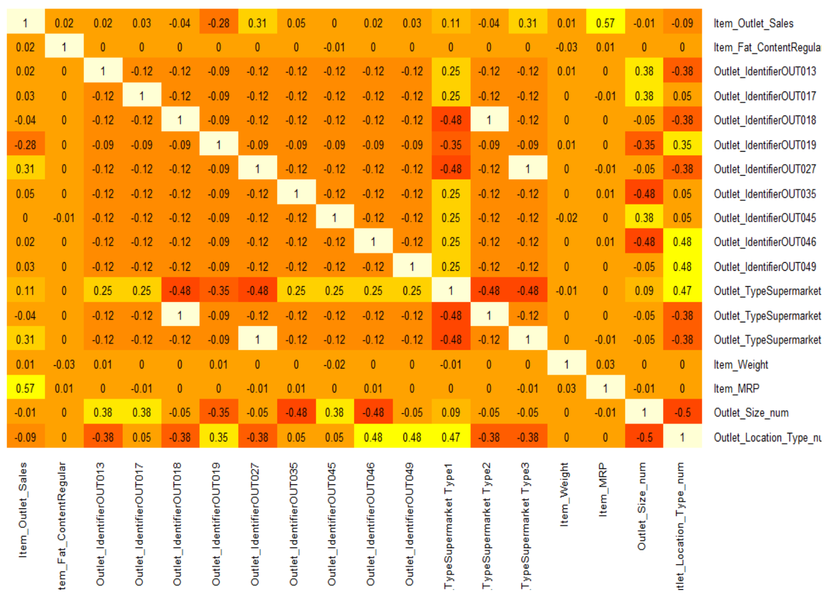
```
test = combi[(nrow(train) + 1):nrow(combi)]
test[,Item_Outlet_Sales := NULL] # removing Item_Outlet_Sales as it contains only
NA for test dataset.
```

#Examining the correlation among variables

```
library('corrplot')
cor_train = cor(train[,-c("Item_Identifier",)]])
```

#Heatmap for Correlation among variables

```
library(gplots)
heatmap.2((cor_train), Rowv = FALSE, Colv = FALSE, dendrogram = "none",
cellnote = round(cor_train,2), notecol = "black", key = FALSE, trace = 'none',
margins = c(10,10)
```



Prediction Model

Linear regression *is the simplest and most widely used statistical technique for predictive modeling. Given below is the linear regression equation:*

y = A + Bx

where ***x*** *is the independent variable, Y is the target variable . Magnitude of a coefficient wrt to the other coefficients determines the importance of the corresponding independent variable.*

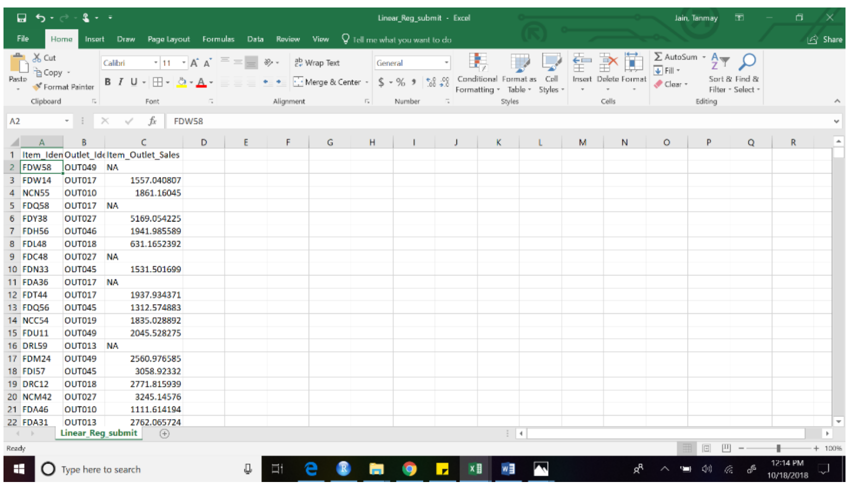
Linear regression on the training dataset

#Applying Prediction Model
 # Linear Regression

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

Applying the predictive model on our test data. Storing the result from our test data in a separate file(final) with a predicted Item_outlet_sales column

```
final$Item_Outlet_Sales = predict(Pred, test[,-c("Item_Identifier")])
write.csv(final, "SampleSubmission_TmnO39y.csv", row.names = F)
```



	Item_Iden	Outlet_Ide	Item_Outlet_Sales
2	FDW58	OUT049	NA
3	FDW14	OUT017	1557.040807
4	NCN55	OUT010	1861.16045
5	FDQ58	OUT017	NA
6	FDY38	OUT027	5169.054225
7	FDH56	OUT046	1941.985589
8	FDL48	OUT018	631.1652392
9	FDC48	OUT027	NA
10	FDN33	OUT045	1531.501699
11	FDA36	OUT017	NA
12	FDT44	OUT017	1937.934371
13	FDQ56	OUT045	1312.574883
14	NCC54	OUT019	1835.028892
15	FDU11	OUT049	2045.528275
16	DRL59	OUT013	NA
17	FDM24	OUT049	2560.976585
18	FDI57	OUT045	3058.92332
19	DRC12	OUT018	2771.815939
20	NCM42	OUT027	3245.14576
21	FDA46	OUT010	1111.614194
22	FDA31	OUT013	2762.065724

XGBoost

Now we can use XGboost on our model to fine tune the model and get the best and accurate results.XGBoost works only with numeric variables and we have already replaced the categorical variables with numeric variables. There are many tuning parameters in XGBoost.For use in XGboost dataset has to be converted into a matrix and then the algorithm is applied.
Let's apply the xgboost algorithm on our training and test data.

#Performing XGboost algrorithm on train and test dataset.

Setting parameter for our 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*. It is a technique which involves reserving a particular sample of a dataset on which you do not train the model. Later, you test your model on this sample before finalizing it.*

We are going to use the xgb.cv() function for cross validation. This function comes with the xgboost package itself.

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)

Applying cross validation with the selected parameters.

```
[1]     train-rmse:2748.400976+8.528842       test-rmse:2748.290088+30.124537
Multiple eval metrics are present. Will use test_rmse for early stopping.
Will train until test_rmse hasn't improved in 30 rounds.

[11]    train-rmse:2551.514258+8.327016       test-rmse:2552.793457+29.442380
[21]    train-rmse:2380.075733+6.001986       test-rmse:2383.249756+30.028548
[31]    train-rmse:2224.724072+5.054612       test-rmse:2229.533398+29.504359
[41]    train-rmse:2081.090869+7.037615       test-rmse:2087.791748+28.504698
[51]    train-rmse:1954.805274+9.736606       test-rmse:1963.424097+25.285533
[61]    train-rmse:1844.952783+11.118738      test-rmse:1855.407275+23.940636
[71]    train-rmse:1745.022656+7.403129       test-rmse:1757.561475+28.680117
[81]    train-rmse:1658.430029+9.543994       test-rmse:1673.158716+28.395715
[91]    train-rmse:1583.468188+11.689022      test-rmse:1600.651147+27.723948
[101]   train-rmse:1516.908814+12.843290      test-rmse:1536.585522+26.325469
[111]   train-rmse:1456.907129+10.194814      test-rmse:1479.279809+26.422530
[121]   train-rmse:1406.177368+11.093508      test-rmse:1431.087988+27.400241
[131]   train-rmse:1358.390112+6.430485       test-rmse:1386.210767+27.446317
```

```
[141]	train-rmse:1316.460181+5.268382	test-rmse:1347.140210+26.427533
[151]	train-rmse:1279.544824+5.138776	test-rmse:1313.027783+26.502236
[161]	train-rmse:1248.949438+3.382555	test-rmse:1285.502637+27.383918
[171]	train-rmse:1220.155615+4.179704	test-rmse:1259.294214+26.095743
[181]	train-rmse:1197.076563+5.224847	test-rmse:1238.998193+24.516842
[191]	train-rmse:1174.581348+5.726388	test-rmse:1219.188892+23.834278
[201]	train-rmse:1154.872436+6.351832	test-rmse:1201.969092+24.620981
[211]	train-rmse:1138.095142+6.938899	test-rmse:1187.641333+23.400480
[221]	train-rmse:1123.213892+6.674663	test-rmse:1175.458691+22.986347
[231]	train-rmse:1109.954126+7.966749	test-rmse:1164.786670+23.577105
[241]	train-rmse:1099.814795+7.970723	test-rmse:1156.671558+23.401788
[251]	train-rmse:1088.750244+8.068297	test-rmse:1148.286279+22.373073
[261]	train-rmse:1079.581079+8.374842	test-rmse:1141.451270+21.846551
[271]	train-rmse:1071.932422+8.098584	test-rmse:1135.881372+20.846314
[281]	train-rmse:1064.592041+7.610936	test-rmse:1130.755053+20.888343
[291]	train-rmse:1057.093506+7.370741	test-rmse:1125.611279+20.708393
[301]	train-rmse:1049.638940+6.947628	test-rmse:1121.030225+21.017280
[311]	train-rmse:1043.770459+7.021429	test-rmse:1117.484107+20.871540
[321]	train-rmse:1038.844898+6.781503	test-rmse:1114.659082+20.847066
[331]	train-rmse:1034.266821+6.230731	test-rmse:1112.041260+20.710601
[341]	train-rmse:1029.090906+6.480299	test-rmse:1109.136743+20.400393
[351]	train-rmse:1025.439319+6.719346	test-rmse:1107.693970+20.322520
[361]	train-rmse:1021.246313+6.543960	test-rmse:1105.446533+19.734214
[371]	train-rmse:1017.543896+6.904457	test-rmse:1103.867993+19.756163
[381]	train-rmse:1013.656152+6.287457	test-rmse:1102.397217+19.562188
[391]	train-rmse:1009.918982+6.335194	test-rmse:1101.052759+19.263717
[401]	train-rmse:1006.594068+6.409144	test-rmse:1099.753638+18.940260
[411]	train-rmse:1003.658337+5.944948	test-rmse:1098.802808+18.899167
[421]	train-rmse:1000.405566+5.919790	test-rmse:1097.726196+18.794580
[431]	train-rmse:997.665357+5.826951	test-rmse:1096.932129+18.683783
[441]	train-rmse:995.202979+5.801526	test-rmse:1096.352734+18.801478
[451]	train-rmse:992.721558+5.528707	test-rmse:1095.741358+18.838021
[461]	train-rmse:990.064819+5.657340	test-rmse:1095.170654+18.711087
[471]	train-rmse:987.155164+5.126352	test-rmse:1094.569556+18.582263
[481]	train-rmse:984.655298+5.140231	test-rmse:1094.054321+18.512455
[491]	train-rmse:982.050720+4.892445	test-rmse:1093.442065+18.555153
[501]	train-rmse:979.757044+4.904511	test-rmse:1093.015137+18.500238
[511]	train-rmse:977.554785+5.049694	test-rmse:1092.725244+18.264919
[521]	train-rmse:975.447400+5.015756	test-rmse:1092.470825+18.233791
[531]	train-rmse:973.504736+5.132971	test-rmse:1092.231299+18.055284
[541]	train-rmse:971.191284+4.879789	test-rmse:1091.908569+17.859081
[551]	train-rmse:969.183777+4.500372	test-rmse:1091.676343+17.902979
[561]	train-rmse:967.273340+4.298480	test-rmse:1091.425830+17.924387
[571]	train-rmse:965.271631+4.453757	test-rmse:1091.266504+17.919829
[581]	train-rmse:963.466052+4.234312	test-rmse:1091.186035+17.858266
[591]	train-rmse:961.560877+4.025592	test-rmse:1091.138819+17.814726
[601]	train-rmse:959.522327+4.049206	test-rmse:1091.051270+17.794350
[611]	train-rmse:957.638757+4.103302	test-rmse:1090.956543+17.701354
[621]	train-rmse:955.717420+3.685886	test-rmse:1090.843774+17.731576
[631]	train-rmse:954.215930+3.654264	test-rmse:1090.933862+17.643958
[641]	train-rmse:952.310083+3.690277	test-rmse:1090.921875+17.573324
[651]	train-rmse:950.461719+3.814913	test-rmse:1090.937134+17.422205
Stopping. Best iteration:
[628]	train-rmse:954.708093+3.689296	test-rmse:1090.816919+17.660791
```

Model Testing using XGBoost and its powers.

As per the above set, we got the best validation/test score at the 628th iteration.
Hence, we will use nrounds = 628 for building the XGBoost model.

#As per the above set, we got the best validation/test score at the 628th iteration. Hence, we will use nrounds = 628 for building the XGBoost model.

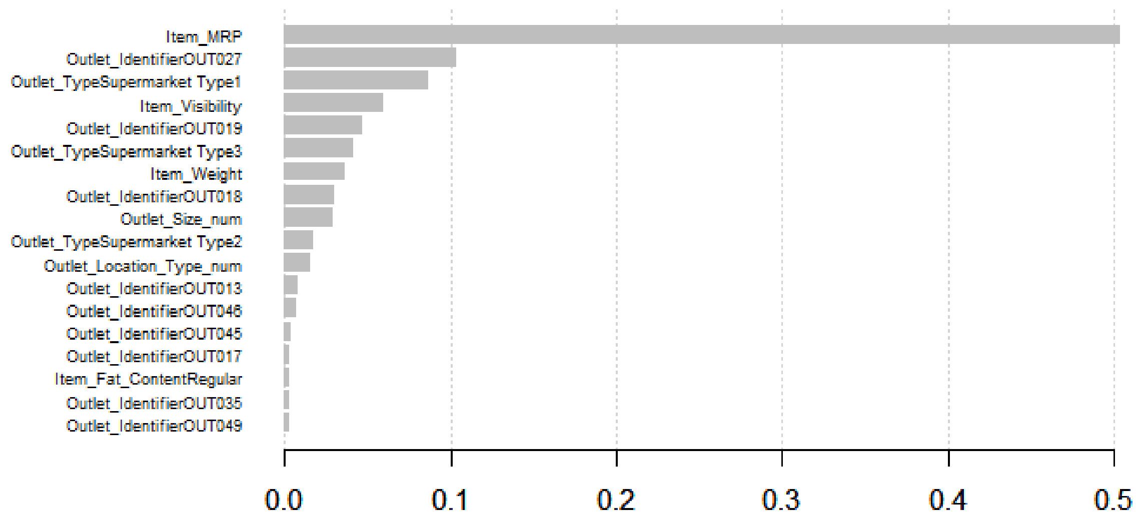
xgb_model = xgb.train(data = dtrain, params = param_list, nrounds = 628)

#Let's use Xgboost importance function to Show Importance Of Features In A Model

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

head(var_imp)

xgb.plot.importance(var_imp)



Item_MRP , OUT027,Supermarket_Type1, Item Visibility tops the chart in importance factor for our model. Item_Outlet_Sales depends upon a lot and is affected significantly by these factors .