Name: chitipolu sri sudheera

Prediction of Sales store wise and Product wise

Bigmart Sales Data Set

Retail is another industry which extensively uses analytics to optimize business processes. Tasks like product placement, inventory management, customized offers, product bundling, etc. are being smartly handled using data science techniques. As the name suggests, this data comprises of transaction records of a sales store. This is a regression problem. The data has 8523 rows of 12 variables.

Problem: Predict the sales of a store.

Problem Statement

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 and find out the sales of each product at a particular store.
Using this model, BigMart will try to understand the properties of products and stores which play a key role in increasing sales.

The Hypotheses

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.

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.

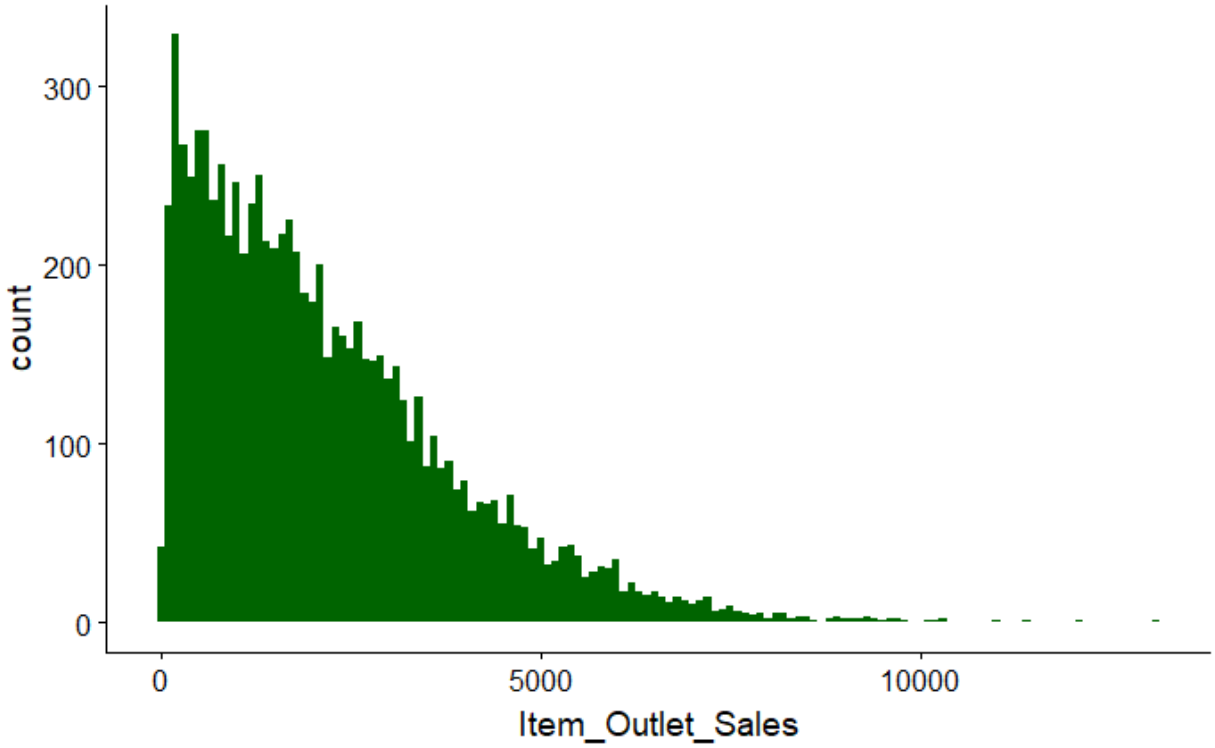
Exploratory Data Analysis (EDA)

Target Variable

Since target variable is continuous, we can visualise it by plotting its histogram.

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

```
  xlab("Item_Outlet_Sales")
```



As you can see, it is a right skewd variable and would need some data transformation to treat its skewness.

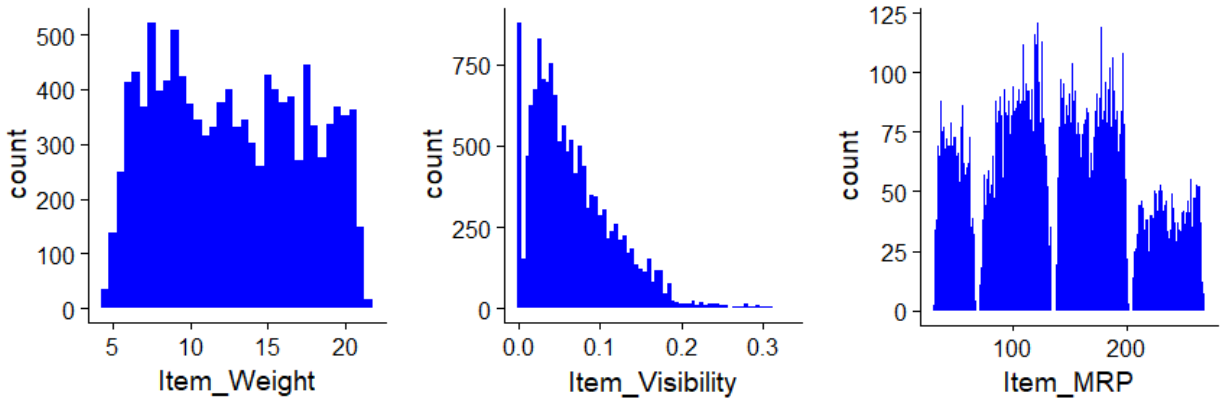
Independent Variables (numeric variables)

Now let's check the numeric independent variables. We'll again use the histograms for visualizations because that will help us in visualizing the distribution of the 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, f
ill = "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
```

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



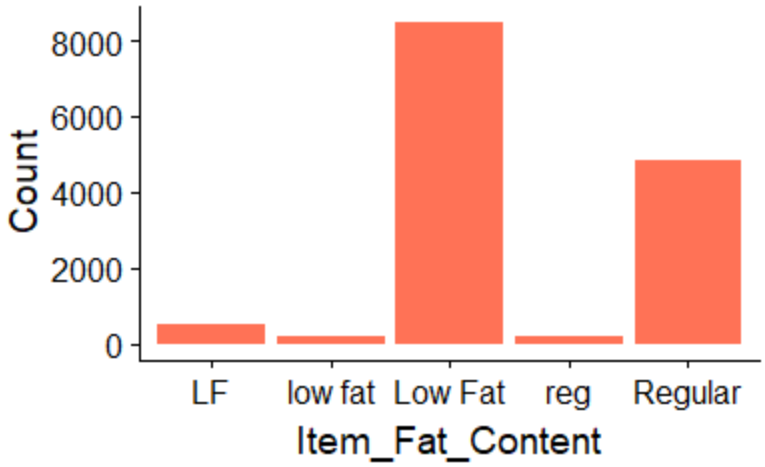
Observations

- 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.

Independent Variables (categorical variables)

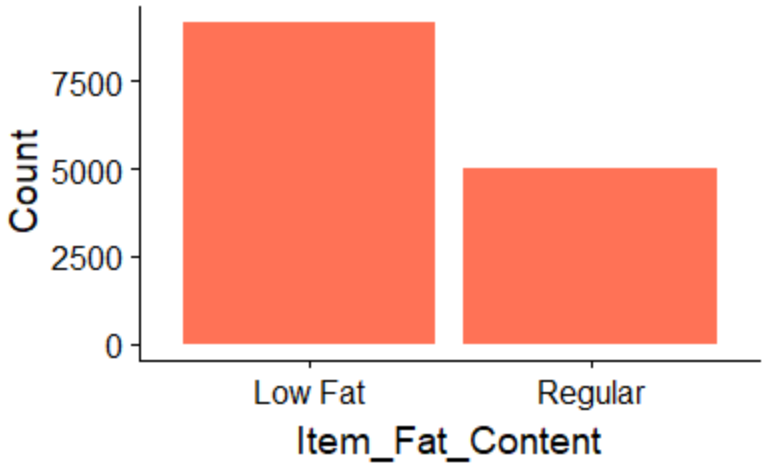
Now we'll try to explore and gain some insights from the categorical variables. A categorical variable or feature can have only a finite set of values. Let's first plot Item_Fat_Content.

```
ggplot(combi %>% group_by(Item_Fat_Content) %>% summarise(Count = n())) +

  geom_bar(aes(Item_Fat_Content, Count), stat = "identity", fill = "coral1")
```



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

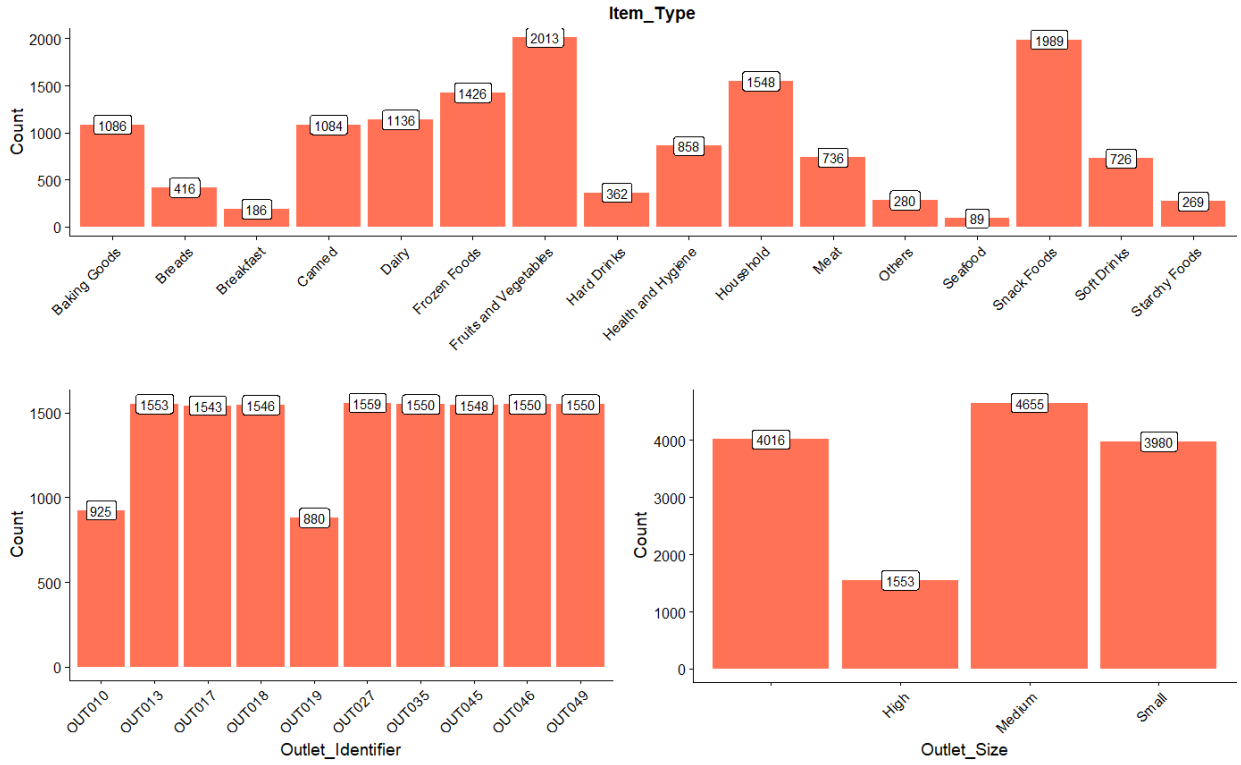
```
combi$Item_Fat_Content[combi$Item_Fat_Content == "LF"] = "Low Fat"

combi$Item_Fat_Content[combi$Item_Fat_Content == "low fat"] = "Low Fat"

combi$Item_Fat_Content[combi$Item_Fat_Content == "reg"] = "Regular"

ggplot(combi %>% group_by(Item_Fat_Content) %>% summarise(Count = n())) +

  geom_bar(aes(Item_Fat_Content, Count), stat = "identity", fill = "coral1")
```



Now let's 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))
```

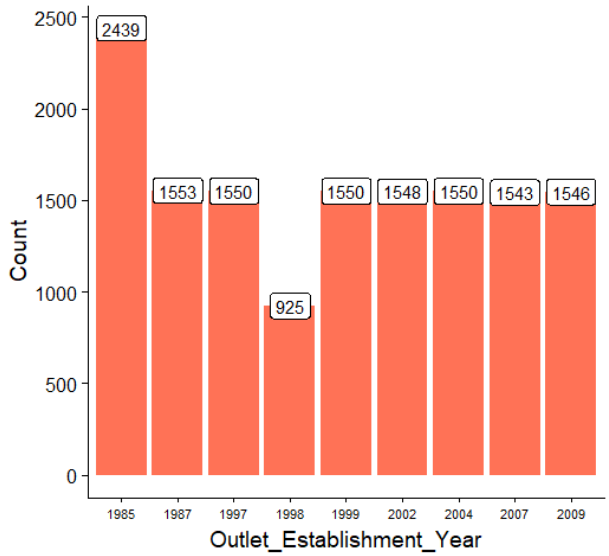
```
# plot for Outlet_Size

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)
```

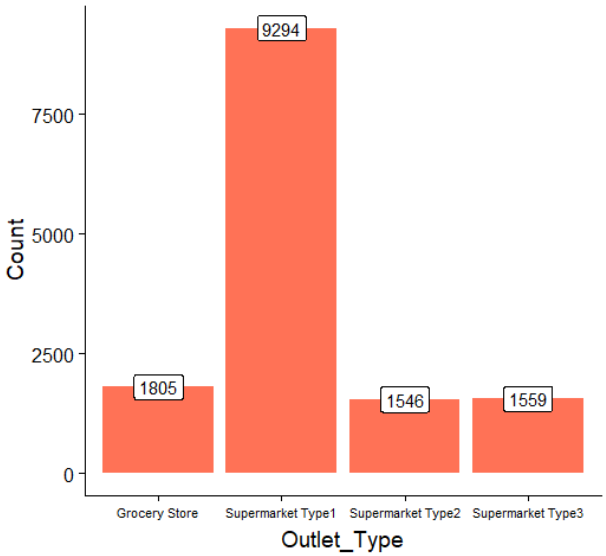


In Outlet_Size's plot, for 4016 observations, Outlet_Size is blank or missing.We will check for this in the bivariate analysis to substitute the missing values in the Outlet_Size.

We'll also check the 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), vj
ust = 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)
```





Observations

- Lesser number of observations in the data for the outlets established in the year 1998 as compared to the other years.
- Supermarket Type 1 seems to be the most popular category of Outlet_Type.

scatter plots for the continuous or numeric variables and **violin plots** for the categorical variables.

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

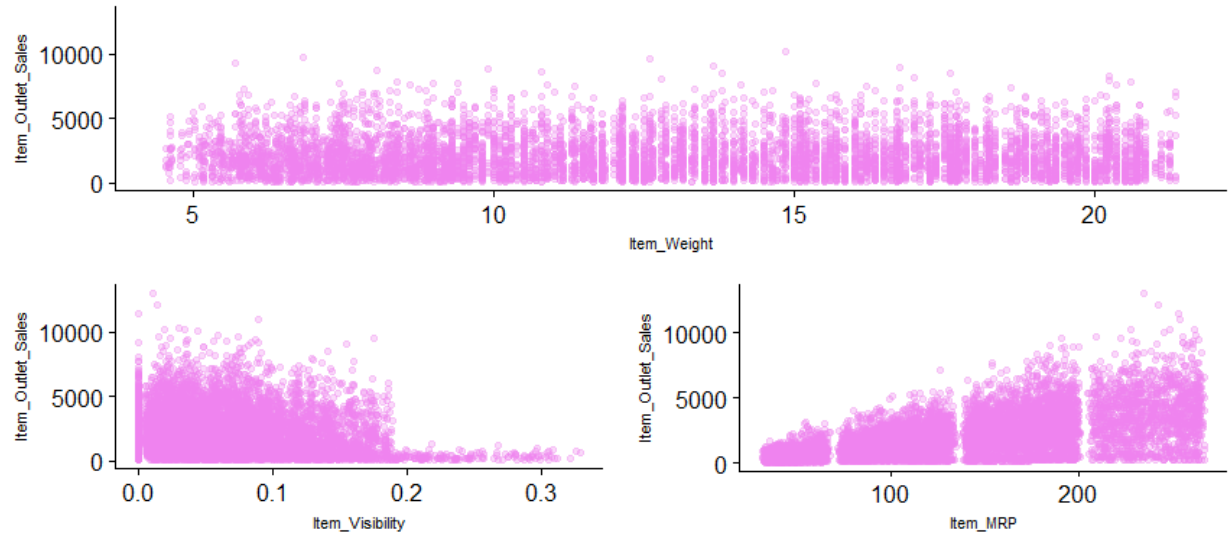
Target Variable vs Independent Numerical Variables

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)
```

```
Removed 1463 rows containing missing values (geom_point).
```



Observations

- Item_Outlet_Sales is spread well across the entire range of the Item_Weight without any obvious 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. We will take note of this issue and deal with it in the later stages.
- 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.

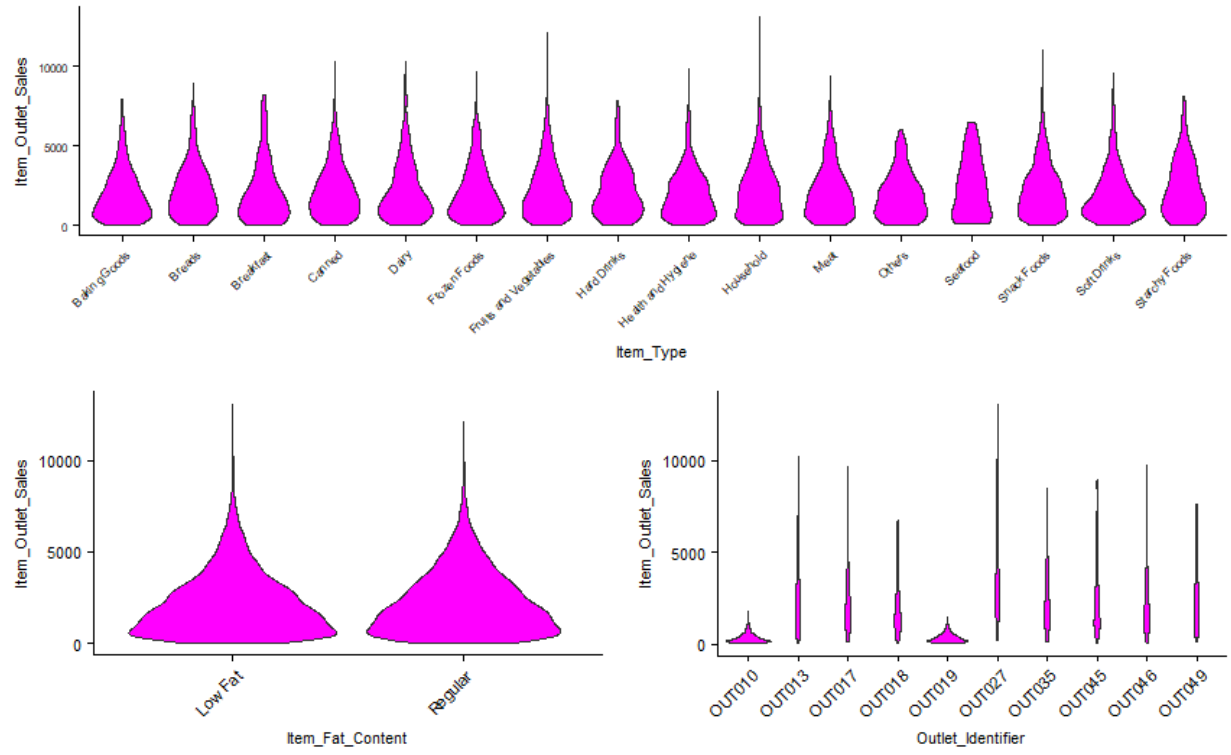
Target Variable vs Independent Categorical Variables

Now we'll visualise 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 boxplots here, but instead we'll use the violin plots as they show the full distribution of the data. The width of a violin plot at a particular level indicates the concentration or density of data at that level. The height of a violin tells us about the range of the target variable values.

```
# 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)
```

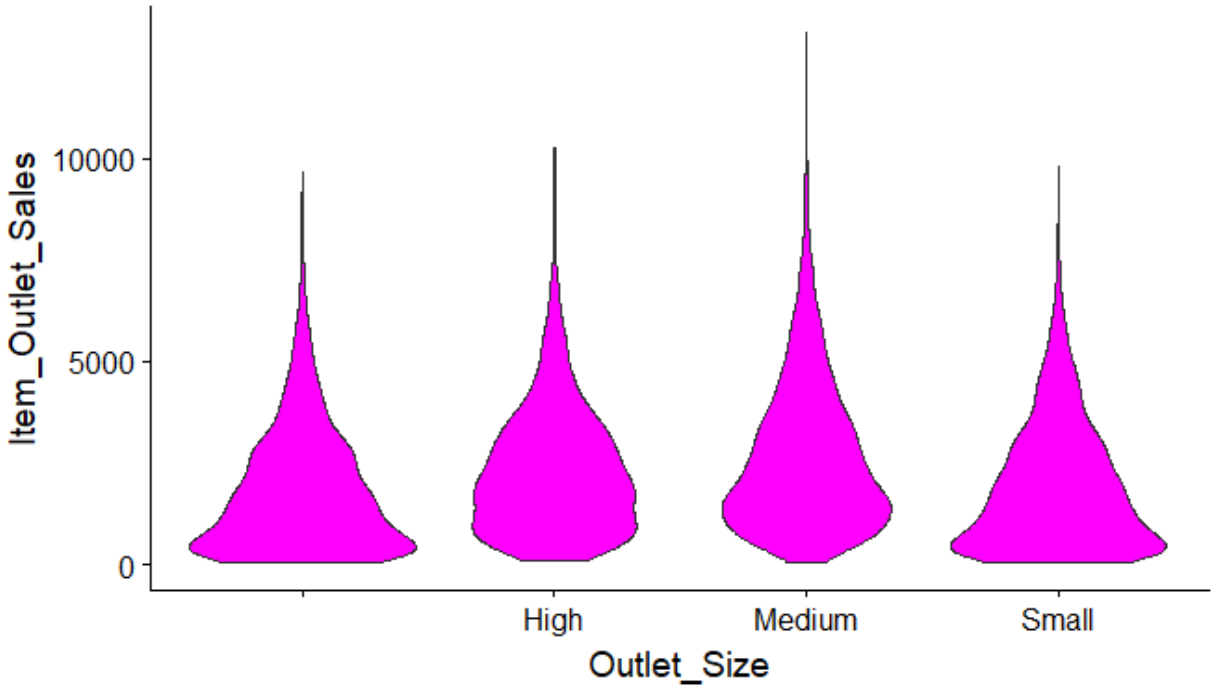


Observations

- 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 different from the rest of the categories of Outlet_Identifier.

In the univariate analysis, we came to know about the empty values in Outlet_Size variable. Let's check the distribution of the target variable across Outlet_Size.

```
ggplot(train) + geom_violin(aes(Outlet_Size, Item_Outlet_Sales), fill = "mage
nta")
```

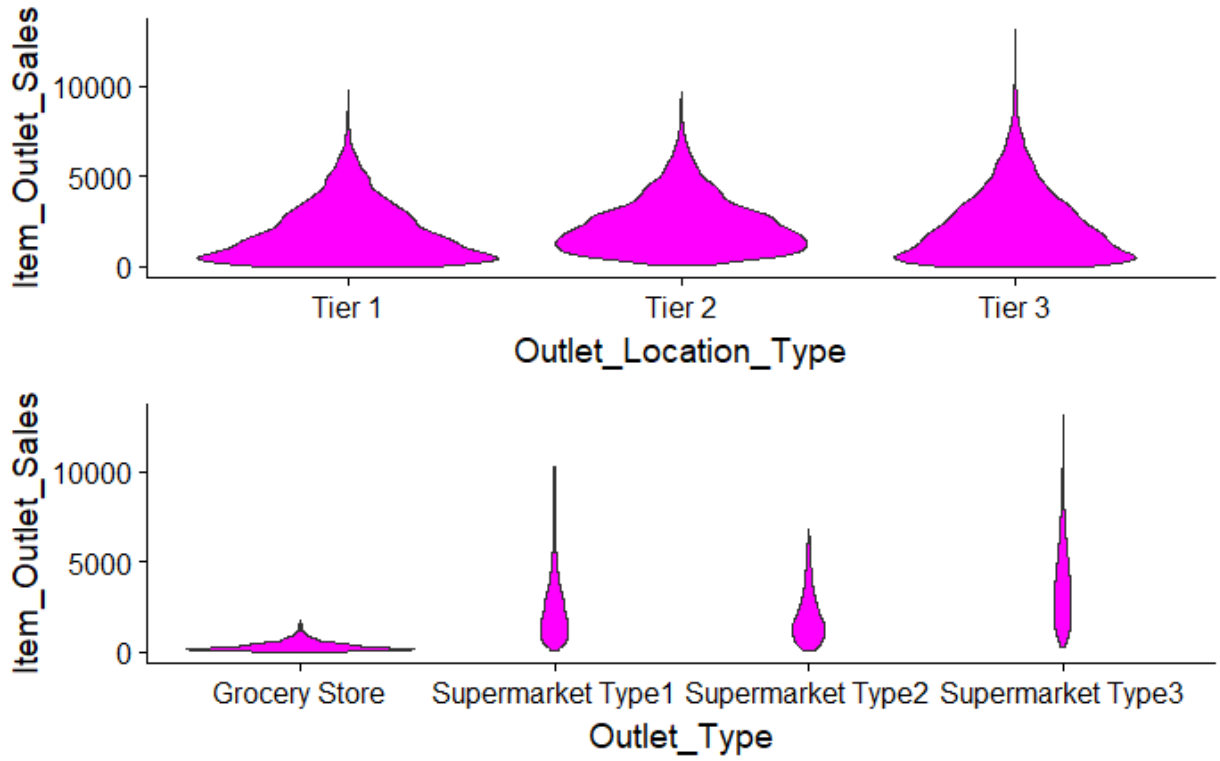


The distribution of 'Small' Outlet_Size is almost identical to the distribution of the blank category (first vioin) of Outlet_Size. So, we can substitute the blanks in Outlet_Size with 'Small'.

Please note that this is not the only way to impute missing values, but for the time being we will go ahead and impute the missing values with 'Small'.

Let's examine the remaining variables.

```
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)
```



Observations

- Tier 1 and Tier 3 locations of Outlet_Location_Type look similar.
- In the Outlet_Type plot, Grocery Store has most of its data points around the lower sales values as compared to the other categories.

These are the kind of insights that we can extract by visualizing our data. Hence, data visualization should be an important part of any kind data analysis.

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

```
[1] 2439
```

Imputing Missing Value

As you can see above, we have missing values in Item_Weight and Item_Outlet_Sales. Missing data in Item_Outlet_Sales can be ignored since they belong to the test dataset. We'll now impute *Item_Weight* with mean weight based on the *Item_Identifier* variable.

```
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)

}
```

Now let's see if there is still any missing data in *Item_Weight*

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

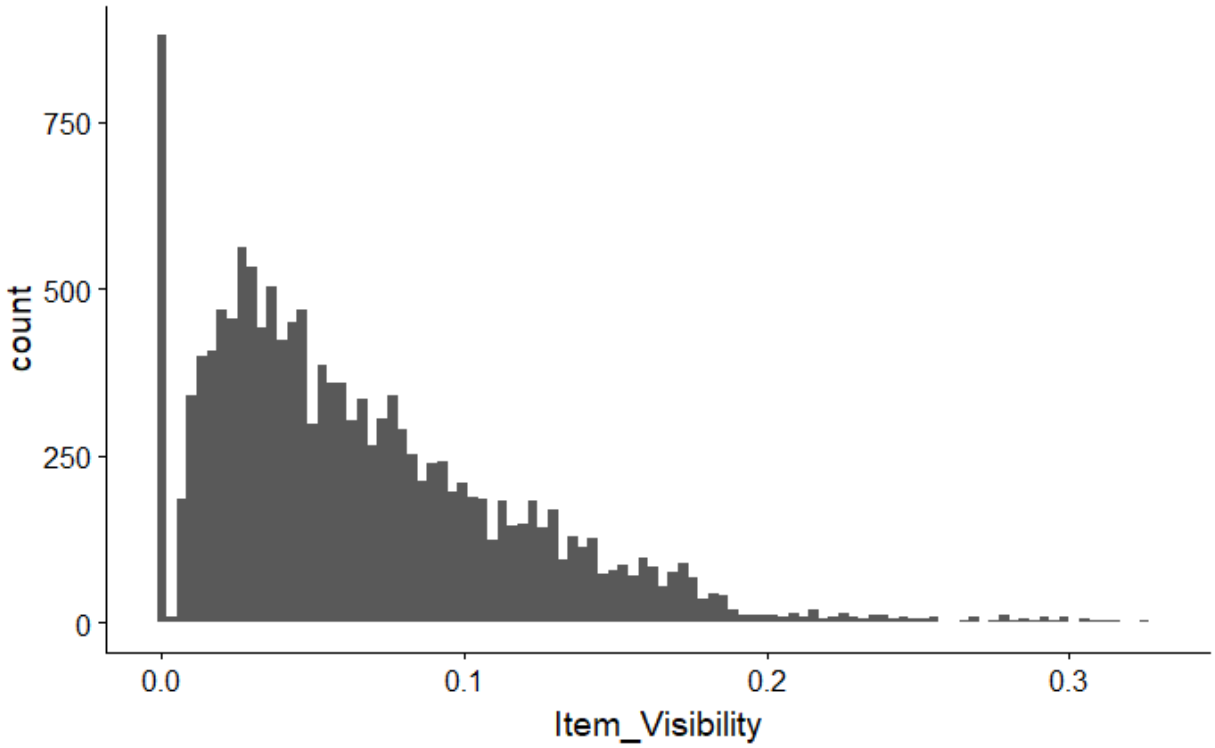
```
[1] 0
```

0 missing values! It means we have successfully imputed the missing data in the feature.

Replacing 0's in Item_Visibility variable

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.

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

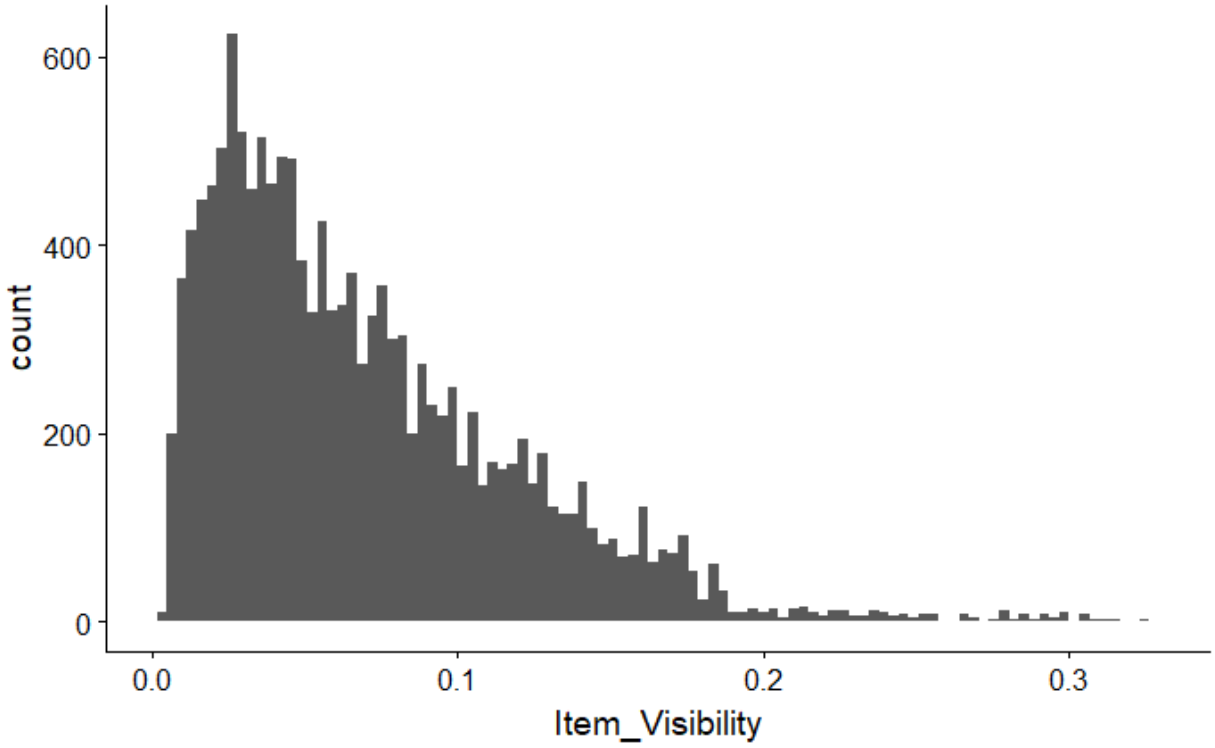


Let's replace the zeroes.

```
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)



}
```

After the replacement of zeroes, We'll plot the histogram of *Item_Visibility* again. In the histogram, we can see that the issue of zero item visibility has been resolved.

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



Feature Engineering

- **Item_Type_new**: Broader categories for the variable *Item_Type*.
- **Item_category**: Categorical variable derived from *Item_Identifier*.
- **Outlet_Years**: Years of operation for outlets.
- **price_per_unit_wt**: *Item_MRP/Item_Weight*
- **Item_MRP_clusters**: Binned feature for *Item_MRP*.

We can have a look at the *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", ifels
e(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))
```

```
                       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
```

Based on the above table we can create a new feature. Let's call it **Item_category**.

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

We will also change the values of *Item_Fat_Content* wherever *Item_category* is 'NC' because non-consumable items cannot have any fat content. We will also create a couple of more features — **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]
```

Earlier in 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, "2
nd",

                                          ifelse(Item_MRP >= 136 & Item_MRP <
203, "3rd", "4th")))]
```

Label encoding for the categorical variables

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

```
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 == "Tie
r 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_Establishme
nt_Year", "Item_Type")], fullRank = T)

ohe_df = data.table(predict(ohe, combi[,-c("Item_Identifier", "Outlet_Establi
shment_Year", "Item_Type")]))

combi = cbind(combi[,"Item_Identifier"], ohe_df)
```

Data PreProcessing

Removing Skewness

```
combi[,Item_Visibility := log(Item_Visibility + 1)] # log + 1 to avoid divisi
on 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 numer
ic 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 for test dataset
```

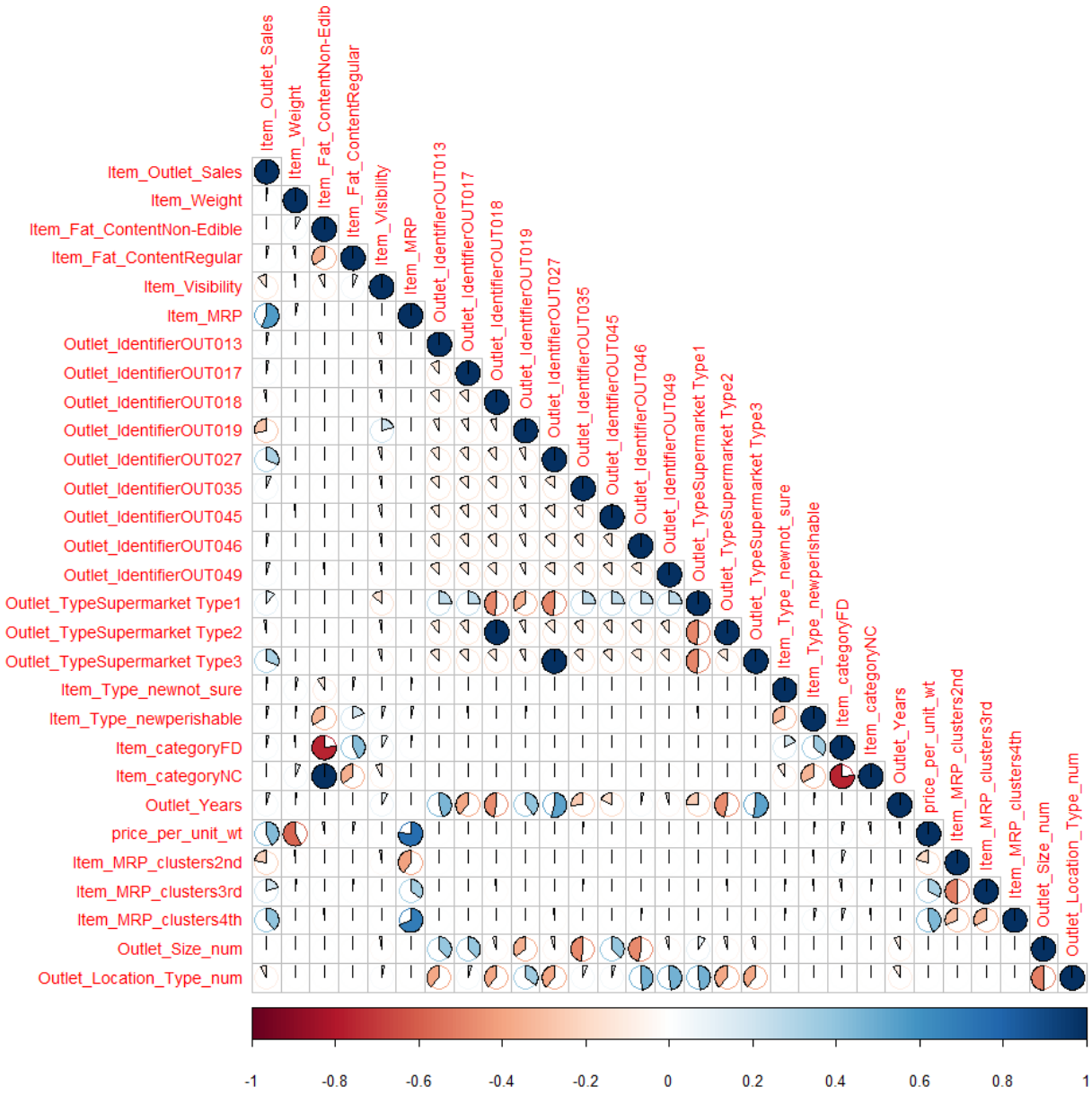
Correlated Variables

Let's examine the correlated features of 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 correlated features if we are using linear regressions.

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

corrplot(cor_train, method = "pie", type = "lower", tl.cex = 0.9)
```



Variables price_per_unit_wt and Item_Weight are highly correlated as the former one was created from the latter. Similarly price_per_unit_wt and Item_MRP are highly correlated for the same reason.

Model Building

- Linear Regression
- Lasso Regression
- Ridge Regression
- RandomForest
- XGBoost

Building Model

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

Making Predictions on test Data

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

submission$Item_Outlet_Sales = predict(linear_reg_mod, test[,-c("Item_Identif
ier")])
```

Leaderboard score: 1202.33

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_S
ales")], y = train$Item_Outlet_Sales,

                      method='glmnet', trControl= my_control, tuneGrid = Gri
d)
```

Mean validation score: 1130.02

Leaderboard score: 1202.26

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_S
ales")], y = train$Item_Outlet_Sales,

                      method='glmnet', trControl= my_control, tuneGrid = Gri
d)
```

Mean validation score: 1135.08

Leaderboard score: 1219.08

Random Forest

```
set.seed(1237)

my_control = trainControl(method="cv", number=5) # 5-fold CV

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")
```

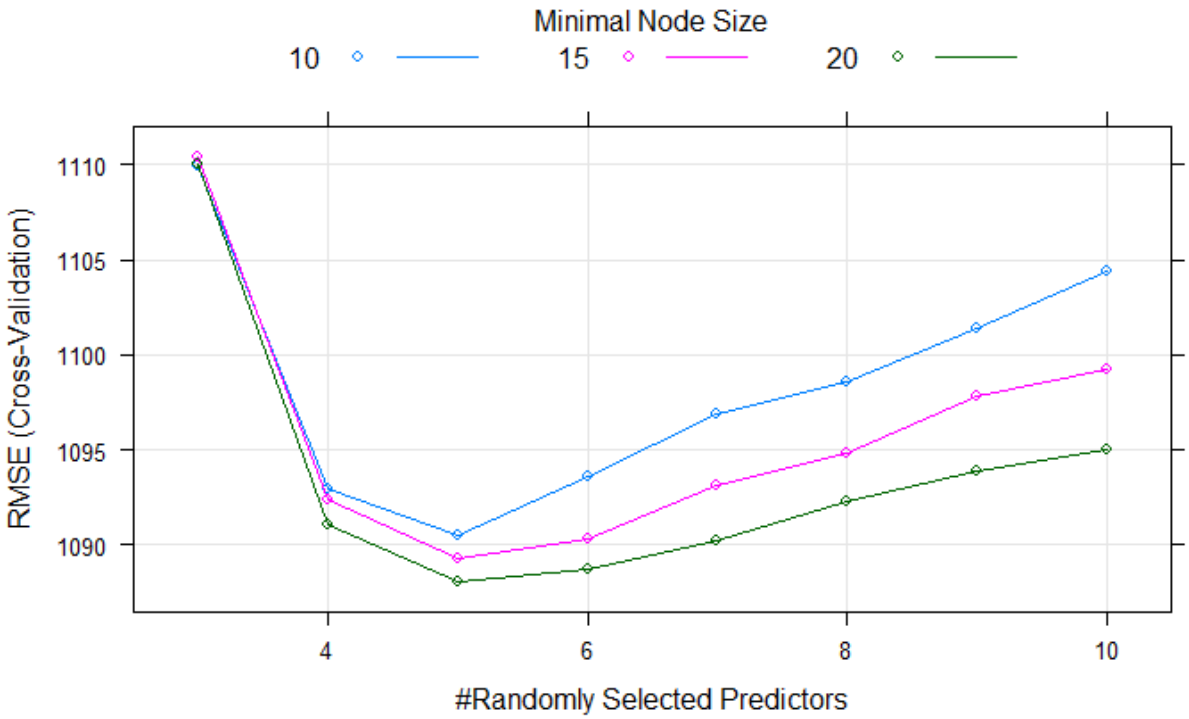
Mean validation score: 1088.05

Leaderboard score: 1157.25

Our score on the leaderboard has improved considerably by using RandomForest. Now let's visualize the RMSE scores for different tuning parameters.

Best Model Parameters

```
plot(rf_mod)
```

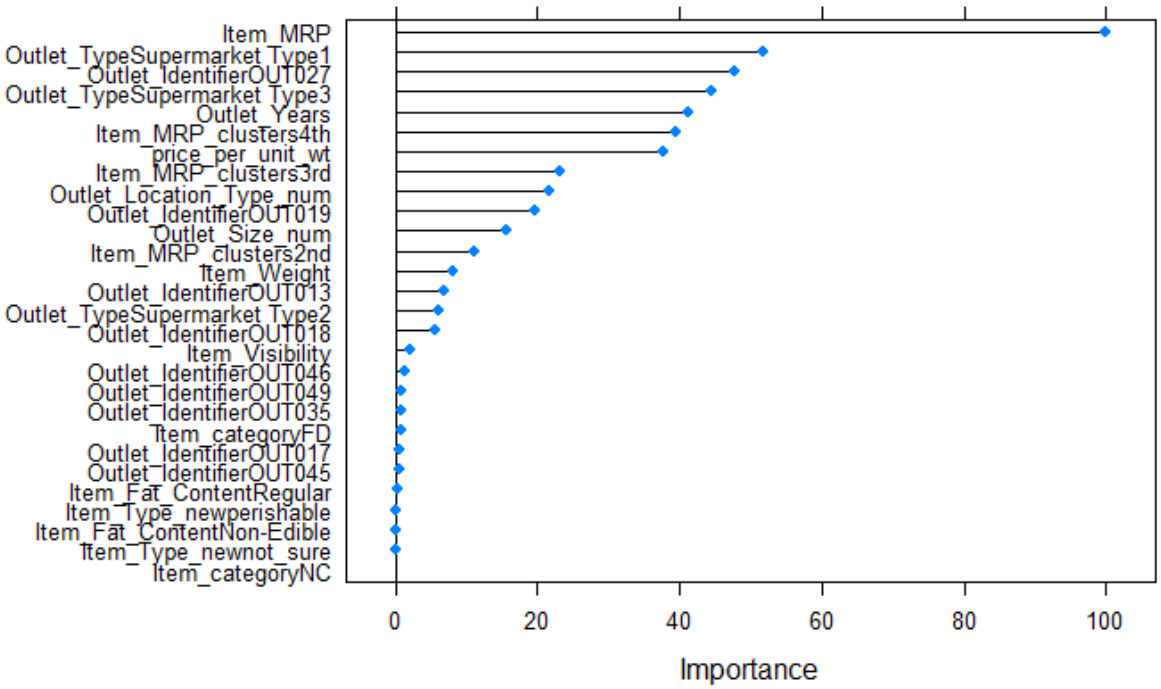


As per the plot shown above, the best score is achieved at mtry = 5 and min.node.size = 20.

Variable Importance

Let's plot feature importance based on the RandomForest model

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



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_Years, Item_MRP_Clusters, are also among the top most important variables. This is why feature engineering plays such a crucial role in predictive modeling.

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_Outl
et_Sales")]), label= train$Item_Outlet_Sales)

dtest = xgb.DMatrix(data = as.matrix(test[,-c("Item_Identifier")]))
```

Cross Validation

We are going to use the xgb.cv() 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.

```
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)
```

```
[1] train-rmse:2746.725586+7.336395 test-rmse:2746.741797+30.998116

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:2537.260938+5.858499 test-rmse:2539.031055+31.050055

[21]    train-rmse:2349.231885+4.482735 test-rmse:2352.873389+32.255671
```

```
[31]    train-rmse:2182.393750+4.819384 test-rmse:2187.824902+31.588163
[41]    train-rmse:2033.432080+3.195964 test-rmse:2040.415454+31.588752
[51]    train-rmse:1900.760058+2.740688 test-rmse:1910.174609+30.442020
[61]    train-rmse:1784.256690+2.877070 test-rmse:1796.284033+28.750707
[71]    train-rmse:1680.479932+2.497050 test-rmse:1695.349048+28.331918
[81]    train-rmse:1590.351367+2.354304 test-rmse:1607.833301+27.908130
[91]    train-rmse:1510.894165+2.035948 test-rmse:1531.593726+27.522115
[101]   train-rmse:1441.319312+1.659380 test-rmse:1465.053589+29.087921
[111]   train-rmse:1381.578052+2.426386 test-rmse:1408.426611+27.691651
[121]   train-rmse:1329.565503+2.454451 test-rmse:1359.327344+27.161061
[131]   train-rmse:1283.176050+2.888625 test-rmse:1316.178931+26.197404
[141]   train-rmse:1243.624048+2.574635 test-rmse:1280.237305+25.250655
[151]   train-rmse:1210.155518+2.584130 test-rmse:1250.134522+25.117012
[161]   train-rmse:1180.639380+2.815261 test-rmse:1224.006177+24.242971
[171]   train-rmse:1155.042627+2.457568 test-rmse:1201.782520+23.791665
[181]   train-rmse:1133.263232+2.564969 test-rmse:1183.207934+22.977824
[191]   train-rmse:1114.468555+2.585744 test-rmse:1167.656421+22.294357
[201]   train-rmse:1098.238403+3.042034 test-rmse:1154.633814+21.729746
[211]   train-rmse:1083.767114+3.474945 test-rmse:1143.161670+21.285152
[221]   train-rmse:1071.447070+3.508526 test-rmse:1134.116235+20.871062
[231]   train-rmse:1060.780518+3.622241 test-rmse:1126.463550+20.575531
[241]   train-rmse:1051.508252+3.255189 test-rmse:1119.907349+20.552631
```

```
[251]   train-rmse:1043.312622+3.268359 test-rmse:1114.685669+20.173664

[261]   train-rmse:1035.746997+2.930902 test-rmse:1110.081641+19.966049

[271]   train-rmse:1028.874536+2.943542 test-rmse:1105.989917+19.530901

[281]   train-rmse:1022.628345+2.937003 test-rmse:1102.838232+19.300798

[291]   train-rmse:1017.216028+2.821218 test-rmse:1100.273462+19.221121

[301]   train-rmse:1012.029614+2.743604 test-rmse:1098.101782+19.030941

[311]   train-rmse:1007.425781+2.755408 test-rmse:1096.336670+18.986031

[321]   train-rmse:1003.200305+2.701284 test-rmse:1094.916968+19.005910

[331]   train-rmse:999.357104+2.599227  test-rmse:1093.706055+19.035905

[341]   train-rmse:995.499194+2.425299  test-rmse:1092.699609+18.929201

[351]   train-rmse:991.902954+2.492520  test-rmse:1091.845703+18.983826

[361]   train-rmse:988.751245+2.519240  test-rmse:1091.190332+18.899727

[371]   train-rmse:985.493445+2.504406  test-rmse:1090.730664+18.812897

[381]   train-rmse:982.394104+2.462072  test-rmse:1090.280884+18.888615

[391]   train-rmse:979.547412+2.462087  test-rmse:1089.998340+18.846508

[401]   train-rmse:976.771387+2.411750  test-rmse:1089.900439+18.755224

[411]   train-rmse:973.951404+2.436468  test-rmse:1089.861890+18.818395

[421]   train-rmse:971.220996+2.519783  test-rmse:1089.869263+18.860485

[431]   train-rmse:968.606470+2.528438  test-rmse:1089.797754+18.815133

[441]   train-rmse:966.202258+2.767049  test-rmse:1089.844604+18.789729

[451]   train-rmse:963.874219+2.858829  test-rmse:1089.895117+18.779128

Stopping. Best iteration:
```

```
[430]   train-rmse:968.889343+2.497103  test-rmse:1089.786206+18.836296
```

Model Training

As per the verbose above, we got the best validation/test score at the 430th iteration. Hence, we will use nrounds = 430 for building the XGBoost model.

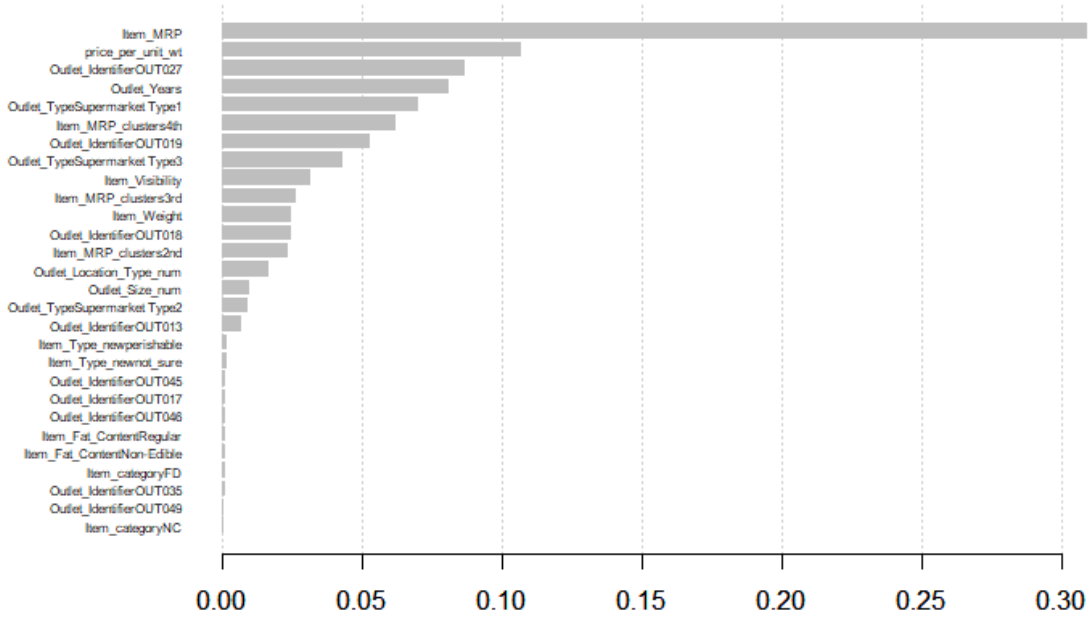
```
xgb_model = xgb.train(data = dtrain, params = param_list, nrounds = 430)
```

Leaderboard score: 1154.70

This model has even outperformed the RandomForest model.

Variable Importance

```
var_imp = xgb.importance(feature_names = setdiff(names(train), c("Item_Identi
fier", "Item_Outlet_Sales")),

                         model = xgb_model)

xgb.plot.importance(var_imp)
```



Again the features created by us, like price_per_unit_wt, Outlet_Years, Item_MRP_Clusters, are among the top most important variables.