AAMD FINAL PROJECT

PREDICTING ITEM SALES

Faseeha Anjum- 17654
Ahmed Raza- 19550
Moeed Zahid- 19061
Rafay Raza- 19347

Instructor: Hassaan Khalid
Institute of Business Administration

Table of Contents

Executive Summary

The product sales at a supermarket can be impacted by a variety of factors including but not limited to outlet type, product MRP etc. This data set consists of sales information for 1559 different products across 10 stores located in various cities for the year 2013. In addition to the sales data, several attributes for each product and store have been defined. The primary objective of this analysis is to build a predictive model that can estimate the sales of each product at a specific outlet. The outlet type in this data set has been divided into 4 categories:

1. SuperMarket Type 1
2. SuperMarket Type 2
3. SuperMarket Type 3
4. Grocery Store

Data Insights

The dataset used for analysis and model development was extracted through the online platform Kaggle. The raw data had the following variable along with a brief description:

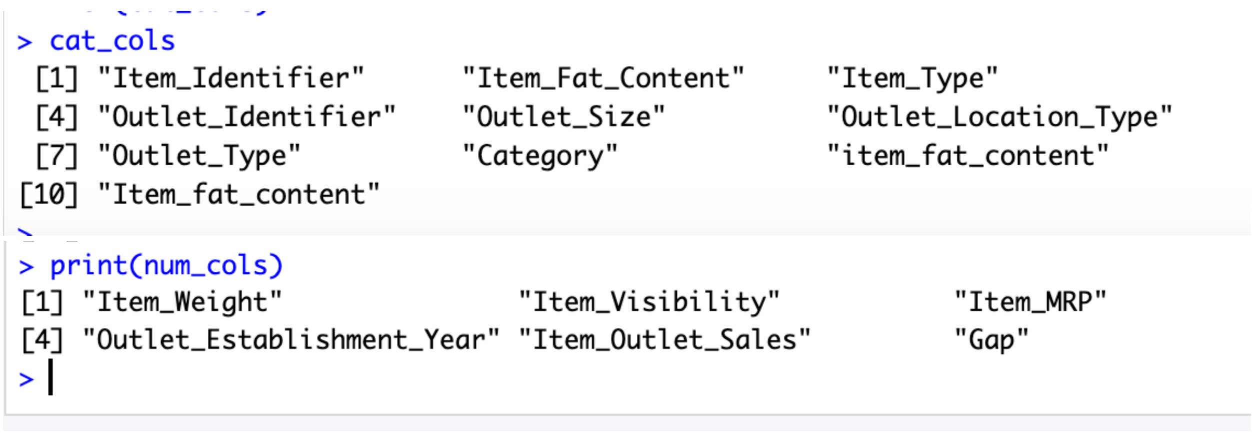
Variable	Description
Item Identifier	Unique product ID
Item Weight	Weight of product
Item Fat Content	Whether the product is low fat or not
Item Visibility	The % of the total display area of all products in a store allocated to the particular product.
Item Type	The category to which the product belongs.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

Assumptions:

In this data set, there is no detailed description provided for the outcome so we assume that the variable ‘Outlet Sales’ represents monthly product sales against each outlet.

Classification of Variables:

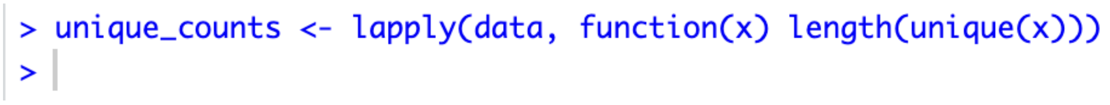
The data set has both categorical and numeric variables. We introduced the values 'cat_cols' and 'num_cols' to represent categorical and numeric variables respectively. The following functions on R were performed to identify the two lists:



```
> cat_cols
 [1] "Item_Identifier"      "Item_Fat_Content"     "Item_Type"
 [4] "Outlet_Identifier"    "Outlet_Size"          "Outlet_Location_Type"
 [7] "Outlet_Type"          "Category"             "item_fat_content"
[10] "Item_fat_content"
```

```
> print(num_cols)
[1] "Item_Weight"                "Item_Visibility"          "Item_MRP"
[4] "Outlet_Establishment_Year" "Item_Outlet_Sales"        "Gap"
>
```

Unique Values were also identified within the variables present in the data set.

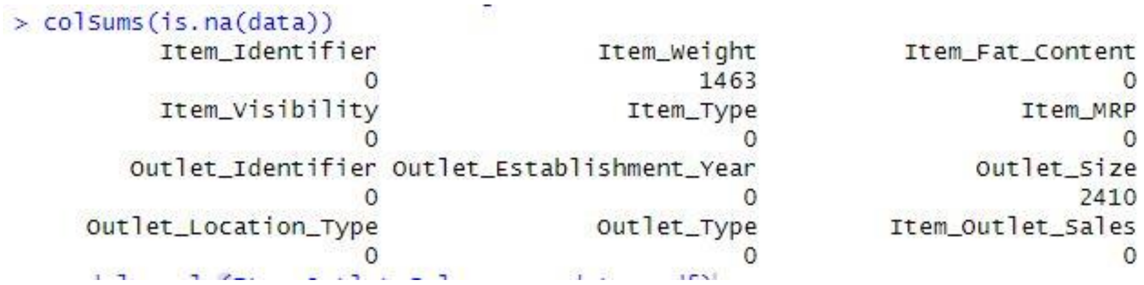


```
> unique_counts <- lapply(data, function(x) length(unique(x)))
>
```

Data Cleaning and Feature Engineering

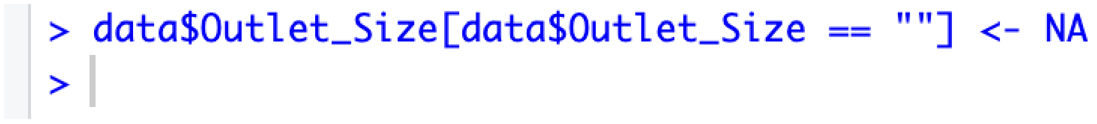
To ensure the accuracy of our predictive model, a series of data cleaning and feature engineering steps were undertaken on the dataset. A overview of the data cleaning and feature engineering process can be divided into the following steps:

Null Values: When checked for null values, there were results within multiple variables in the data set.



```
> colSums(is.na(data))
      Item_Identifier               Item_Weight          Item_Fat_Content
                    0                      1463                         0
      Item_Visibility                 Item_Type                  Item_MRP
                    0                         0                         0
    Outlet_Identifier Outlet_Establishment_Year               Outlet_Size
                    0                         0                      2410
 Outlet_Location_Type               Outlet_Type         Item_Outlet_Sales
                    0                         0                         0
```

Any null value present was denoted by NA except in the column ‘Outlet size’ where missing values were interpreted as blank spaces “ ”. In order to fix this, the blank space was replaced as NA with the help of the following code so that all null values could be treated accurately:



```
> data$Outlet_Size[data$Outlet_Size == ""] <- NA
> 
```

Once the NAs were identified, the values in the categorical variables were replaced by the respective modes and values within the numeric variables were replaced by the respective means.

Duplicate Values: The data set was checked for any duplicate rows but the code resulted in FALSE.

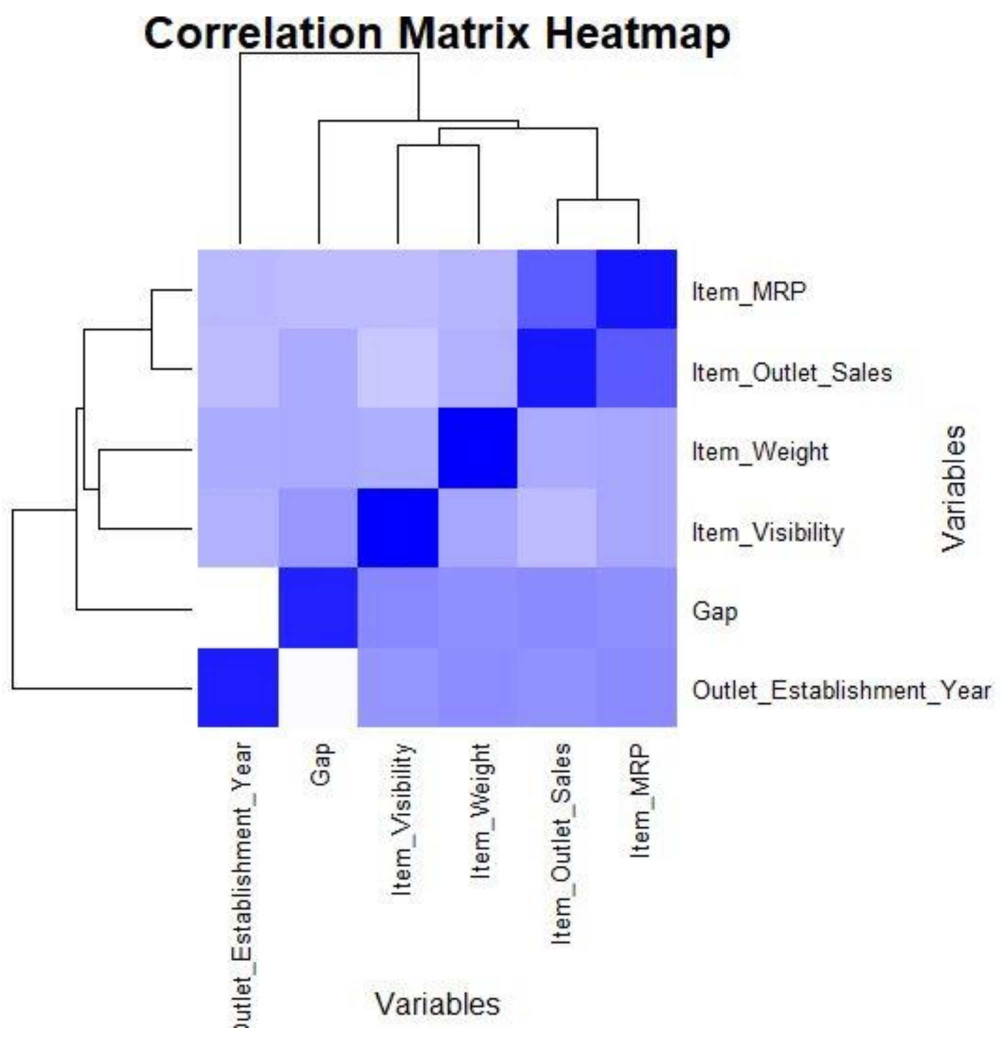
Replacing Values: The variable columns ‘Item_Visibility’ and ‘Item_Fat_Content’ had non-uniform repetitive values. The 0s in Item Visibility were replaced with the mean values of the variable and the levels of Item Fat Content were set uniformly as ‘Low Fat’ and ‘Regular’.

Introducing new variables: The columns ‘Category’ and ‘Gap’ were introduced into the data set:

1. Category: This categorizes the types of items into ‘Drinkables’, ‘Eatables’ and ‘Non-Consumable’
2. Gap: This identifies how old the outlet is (as of 2013).

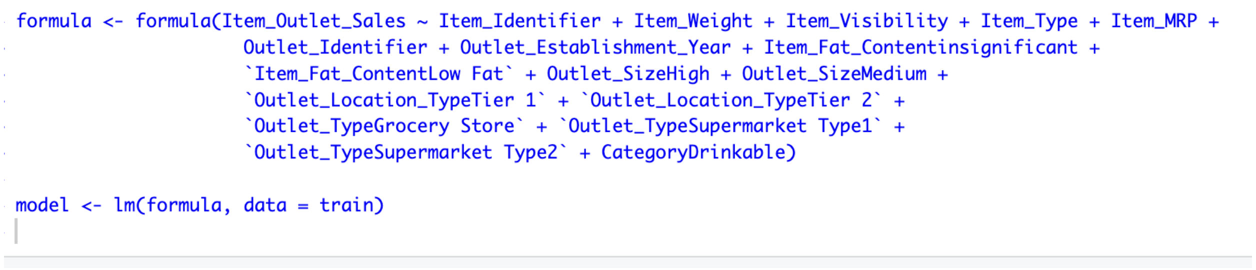
Model Development

Step 1: Correlation Matrix was developed to identify the significant correlations amongst the independent variables and against the dependent variable (Item Outlet Sales).



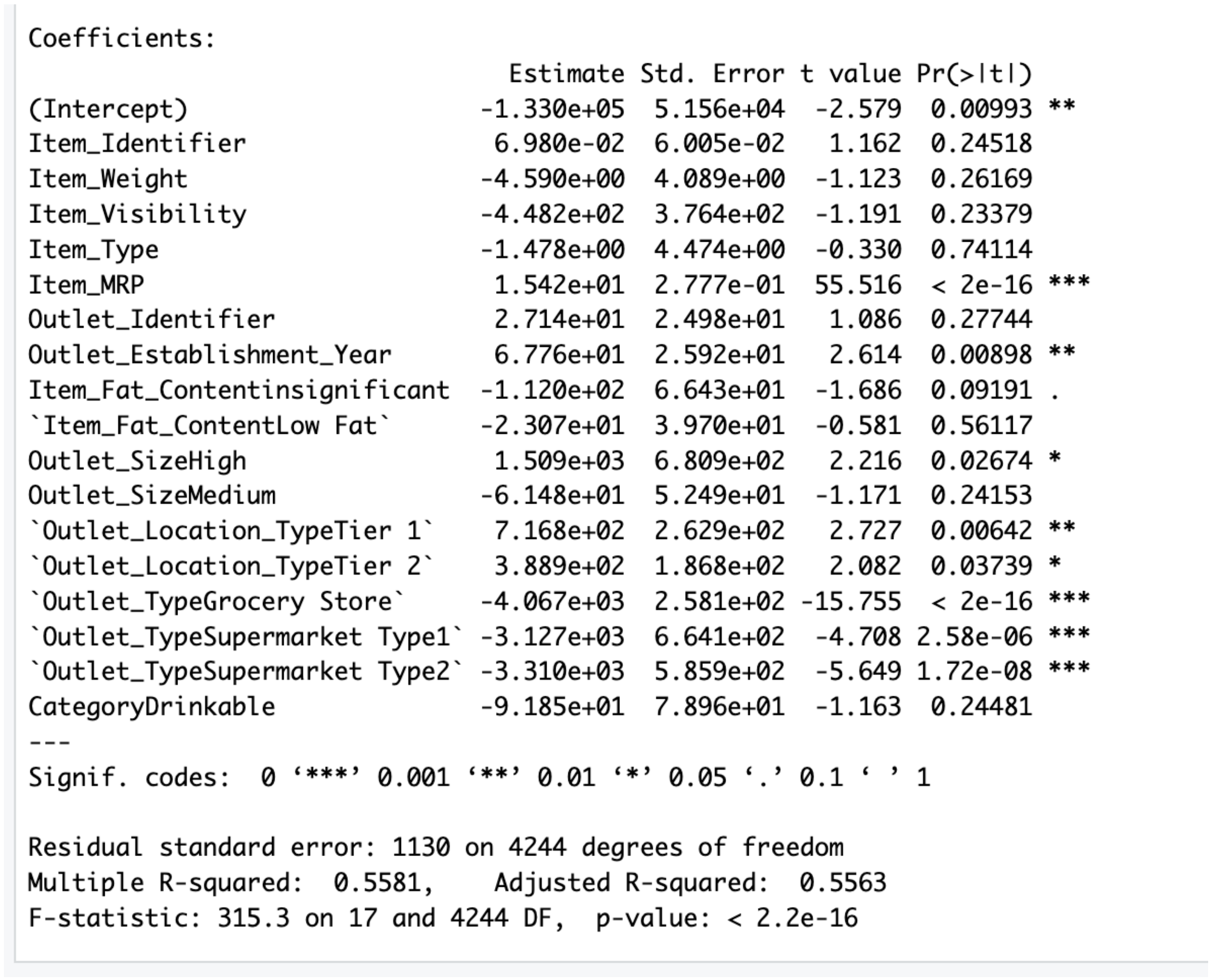
Step 2: The data set was split into testing and training sets with 4261 observations in the testing data set and 4262 observations in the training data set.

Step 3: Linear Regression was applied on the model using the following function:



```
formula <- formula(Item_Outlet_Sales ~ Item_Identifier + Item_Weight + Item_Visibility + Item_Type + Item_MRP +
                     Outlet_Identifier + Outlet_Establishment_Year + Item_Fat_Contentinsignificant +
                     `Item_Fat_ContentLow Fat` + Outlet_SizeHigh + Outlet_SizeMedium +
                     `Outlet_Location_TypeTier 1` + `Outlet_Location_TypeTier 2` +
                     `Outlet_TypeGrocery Store` + `Outlet_TypeSupermarket Type1` +
                     `Outlet_TypeSupermarket Type2` + CategoryDrinkable)

model <- lm(formula, data = train)
```

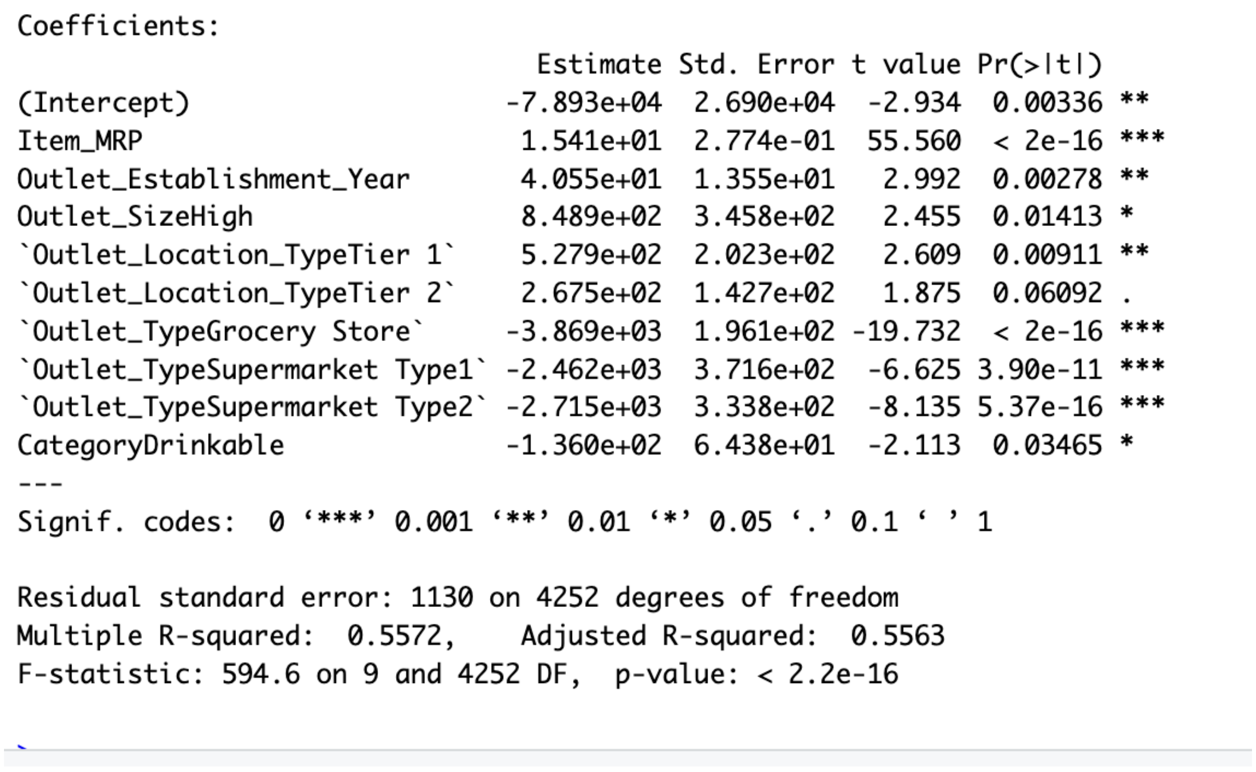
Regression included all the independent variables in the data set against the dependent variable to check for significance levels.



```
Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)                    -1.330e+05  5.156e+04  -2.579  0.00993 ** 
Item_Identifier                 6.980e-02  6.005e-02   1.162  0.24518    
Item_Weight                    -4.590e+00  4.089e+00  -1.123  0.26169    
Item_Visibility                -4.482e+02  3.764e+02  -1.191  0.23379    
Item_Type                      -1.478e+00  4.474e+00  -0.330  0.74114    
Item_MRP                        1.542e+01  2.777e-01  55.516  < 2e-16 ***
Outlet_Identifier               2.714e+01  2.498e+01   1.086  0.27744    
Outlet_Establishment_Year       6.776e+01  2.592e+01   2.614  0.00898 ** 
Item_Fat_Contentinsignificant  -1.120e+02  6.643e+01  -1.686  0.09191 .  
`Item_Fat_ContentLow Fat`      -2.307e+01  3.970e+01  -0.581  0.56117    
Outlet_SizeHigh                 1.509e+03  6.809e+02   2.216  0.02674 *  
Outlet_SizeMedium              -6.148e+01  5.249e+01  -1.171  0.24153    
`Outlet_Location_TypeTier 1`    7.168e+02  2.629e+02   2.727  0.00642 ** 
`Outlet_Location_TypeTier 2`    3.889e+02  1.868e+02   2.082  0.03739 *  
`Outlet_TypeGrocery Store`     -4.067e+03  2.581e+02 -15.755  < 2e-16 ***
`Outlet_TypeSupermarket Type1` -3.127e+03  6.641e+02  -4.708 2.58e-06 ***
`Outlet_TypeSupermarket Type2` -3.310e+03  5.859e+02  -5.649 1.72e-08 ***
CategoryDrinkable              -9.185e+01  7.896e+01  -1.163  0.24481    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1130 on 4244 degrees of freedom
Multiple R-squared:  0.5581,	Adjusted R-squared:  0.5563 
F-statistic: 315.3 on 17 and 4244 DF,  p-value: < 2.2e-16
```

Step 4: Another model was created with the help of Stepwise regression through the following function:

```
> stepmodel<-step(model)
```



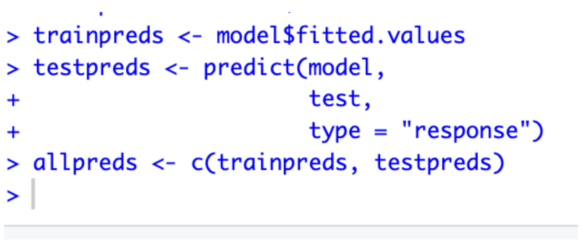
```
Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)                     -7.893e+04  2.690e+04  -2.934  0.00336 ** 
Item_MRP                         1.541e+01  2.774e-01  55.560  < 2e-16 ***
Outlet_Establishment_Year        4.055e+01  1.355e+01   2.992  0.00278 ** 
Outlet_SizeHigh                  8.489e+02  3.458e+02   2.455  0.01413 *  
`Outlet_Location_TypeTier 1`     5.279e+02  2.023e+02   2.609  0.00911 ** 
`Outlet_Location_TypeTier 2`     2.675e+02  1.427e+02   1.875  0.06092 .  
`Outlet_TypeGrocery Store`      -3.869e+03  1.961e+02 -19.732  < 2e-16 ***
`Outlet_TypeSupermarket Type1` -2.462e+03  3.716e+02  -6.625 3.90e-11 ***
`Outlet_TypeSupermarket Type2` -2.715e+03  3.338e+02  -8.135 5.37e-16 ***
CategoryDrinkable               -1.360e+02  6.438e+01  -2.113  0.03465 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1130 on 4252 degrees of freedom
Multiple R-squared:  0.5572,	Adjusted R-squared:  0.5563 
F-statistic: 594.6 on 9 and 4252 DF,  p-value: < 2.2e-16
```

The following independant variables were considered insiginifcant and dropped for the final model:

1. Item Identifier
2. Item Weight
3. Item Visibility
4. Item Type
5. Outlet Identifier
6. Item Fat Content insignificant
7. Item Fat Content Low Fat
8. Outlet Size Medium

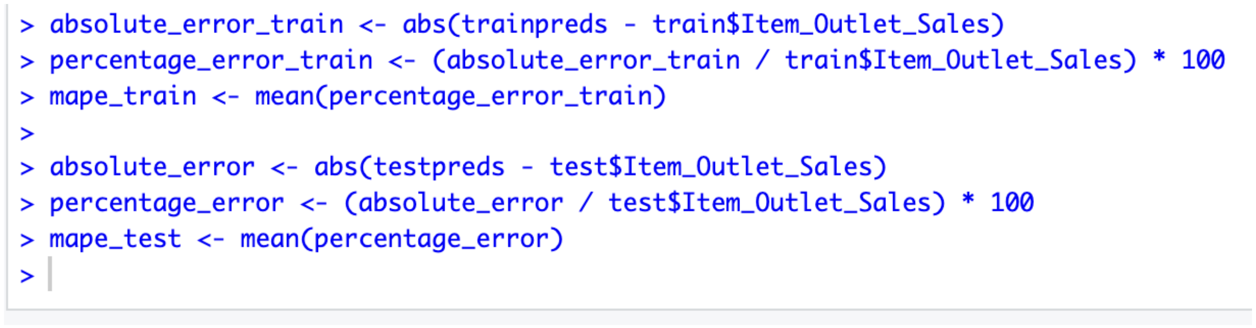
Step 5: Mape Calculation

The split data from testing and training was used to calculate test predictions and train predictions in order to calculate MAPE values.

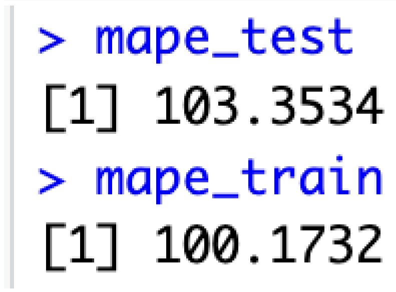


```
> trainpreds <- model$fitted.values
> testpreds <- predict(model,
+                      test,
+                      type = "response")
> allpreds <- c(trainpreds, testpreds)
> 
```

After calculating for absolute error and % error, the following functions were run on R to create values for MAPE of testing and training data:



```
> absolute_error_train <- abs(trainpreds - train$Item_Outlet_Sales)
> percentage_error_train <- (absolute_error_train / train$Item_Outlet_Sales) * 100
> mape_train <- mean(percentage_error_train)
>
> absolute_error <- abs(testpreds - test$Item_Outlet_Sales)
> percentage_error <- (absolute_error / test$Item_Outlet_Sales) * 100
> mape_test <- mean(percentage_error)
> 
```



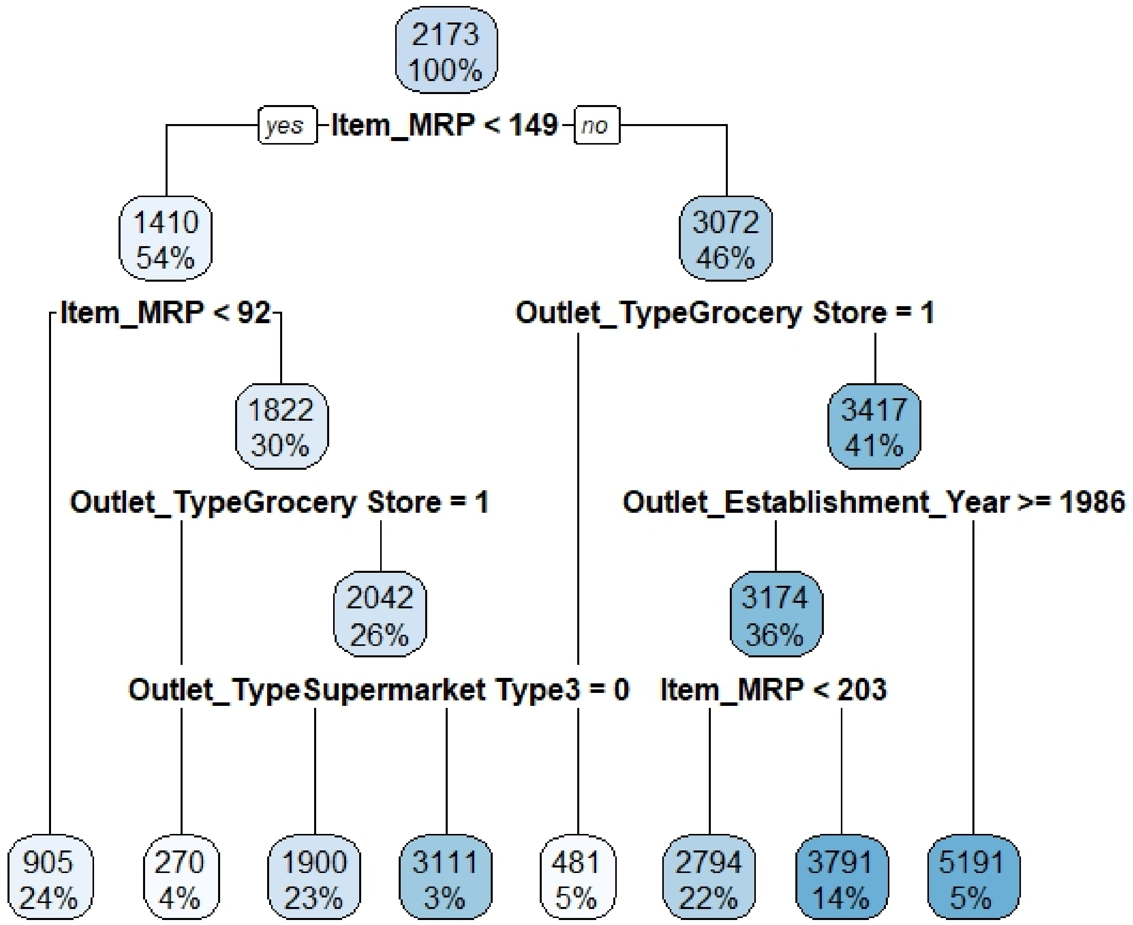
```
> mape_test
[1] 103.3534
> mape_train
[1] 100.1732
```

Decision Tree Regression

A decision tree regression was further carried out on the training data set. Item Outlet Sales was considered as the y variable. This means that the different independent variables were used to predict and reach a conclusion regarding the dependent variable which is, Item Outlet Sales. According to the decision tree below the root node is Item MRP. The total observations are 4262 in our training data. The predicted value of item outlet sales is 2173 in the root node. There are a total of 8 terminal nodes in our decision tree. Terminal nodes are basically those nodes that cannot be split further.

The most important independent variable is Item MRP. It is at the top of the decision tree which splits the model into its different branches. If Item MRP is less than 149, the next branch is whether the Item MRP is less than 92. If it is less than 92 then the predicted item outlet sales are 905. Another example is that if the Item MRP is greater than 149 and the Outlet Type Grocery Store is greater than or equal to 0.5, then the predicted Item Outlet Sales are 481.

The deviance given in the decision tree output is a measure of the impurity within the node. As we move down the decision tree, the accuracy of predicitng item outlet sales improves as the value of deviance goes down.



```
n= 4262

node), split, n, deviance, yval
      * denotes terminal node

 1) root 4262 12255510000 2172.9700
   2) Item_MRP< 149.4721 2305  2406579000 1409.5690
     4) Item_MRP< 91.8817 1037   424030600  904.8299 *
     5) Item_MRP>=91.8817 1268  1502303000 1822.3560
      10) Outlet_TypeGrocery Store>=0.5 157     3816822  270.3275 *
      11) Outlet_TypeGrocery Store< 0.5 1111  1066864000 2041.6790
        22) Outlet_TypeSupermarket Type3< 0.5 981   746520500 1899.9330 *
        23) Outlet_TypeSupermarket Type3>=0.5 130   151896100 3111.3190 *
   3) Item_MRP>=149.4721 1957  6923434000 3072.1220
     6) Outlet_TypeGrocery Store>=0.5 230    17837320  481.0752 *
     7) Outlet_TypeGrocery Store< 0.5 1727  5155844000 3417.1950
      14) Outlet_Establishment_Year>=1986 1519  3692501000 3174.2880
        28) Item_MRP< 203.0361 939  1549976000 2793.5630 *
        29) Item_MRP>=203.0361 580  1786059000 3790.6690 *
      15) Outlet_Establishment_Year< 1986 208   719183300 5191.1150 *
```

XG Boost

Step 1:

```
# Install and load the necessary libraries
install.packages("xgboost")
library(xgboost)
```

Step 2:

```
# Prepare the data
train_matrix <- xgb.DMatrix(as.matrix(train[, -c(which(names(train) == "Item_Outlet_Sales")
test_matrix <- xgb.DMatrix(as.matrix(test[, -c(which(names(test) == "Item_Outlet_Sales"))])
```

Step 3:

```
# Set the XGBoost parameters
params <- list(
  objective = "reg:squarederror",   # Regression objective
  eval_metric = "rmse",             # Evaluation metric: Root Mean Squared Error
  max_depth = 6,                    # Maximum tree depth
  eta = 0.3,                        # Learning rate
  nrounds = 100                     # Number of boosting rounds
)
```

Step 4:

```
# Train the XGBoost model
xgb_model <- xgboost(params = params, data = train_matrix, nrounds = params$nrounds)
```

Step 5:

```
importance <- xgb.importance(model = xgb_model)
```

Feature	Gain	Cover	Frequency
Item_MRP	0.473227	0.341097	0.2577492
Outlet_TypeGrocery Store	0.209442	0.014425	0.0085837
Outlet_Establishment_Year	0.101853	0.027508	0.0469719
Item_Visibility	0.072441	0.235523	0.2081545
Item_Identifier	0.05141	0.177971	0.191464
Item_Weight	0.034166	0.12515	0.1063424
Outlet_Identifier	0.018811	0.018431	0.0674773
Item_Type	0.016915	0.026178	0.0491178
Outlet_Location_TypeTier 2	0.003889	0.000819	0.010968
Outlet_SizeMedium	0.003461	0.001326	0.0104912
Outlet_Location_TypeTier 3	0.002755	0.003735	0.0085837
Outlet_SizeSmall	0.002388	0.004007	0.0078684
Item_Fat_ContentRegular	0.001918	0.001817	0.0042918
Outlet_SizeHigh	0.001629	0.00156	0.0042918
Outlet_TypeSupermarket Type1	0.001473	0.004782	0.0045303
Outlet_TypeSupermarket Type3	0.001358	0.009479	0.0040534
Item_Fat_ContentLow Fat	0.001137	0.000653	0.0047687
CategoryEatables	0.001054	0.000141	0.0016691
CategoryDrinkable	0.000587	0.005239	0.0016691
Outlet_Location_TypeTier 1	8.65E-05	0.000159	0.0009537

XG boost model provides us with feature specific results. This indicates that which feature had a significant effect on sales in the mart. According to the table Item MRP which showed the highest gain had the most significant effect on sale of these marts.

Model Comparison

Comparison	Stepwise Linear Regression		Decision Tree	XgBoost
	Train	Test		
MAPE	100.22	103.3	83.5635	103.3534
R^2	0.5563		0.557	0.502

Stepwise linear has a MAPE of 100.22% on Training Data and 103% on Testing Data whereas the Decision Tree has the Mape of 83% and XgBoost 103%. Decision Tree outperforms both Stepwise Linear Regression and XgBoost which indicates that Decision Tree indicates better accuracy in predicting the targeted variable. Stepwise Linear Regression and Decision has almost the same R^2 Pertaining similar fit to the training data. Therefore, depending on the specific requirements and priorities of the problem at hand, one may choose the model that best balances the trade-off between accuracy (MAPE) and the ability to explain the target variable's variance (R^2).

Suggestions:

The analysis and model show promise but have some clear limitations and opportunities for improvement. The data itself could be expanded and enhanced to be more broadly representative of all categories and distributions. Additional preprocessing of the data could also help, through techniques like adding new columns or transforming non-normal features.

The model's performance is currently hindered by its high error rate and low accuracy score. Tuning the model's hyperparameters may help, but improving the data itself is likely to have an even bigger impact. The model works as a starting point, but still has a long way to go to reach its full potential. Overall, while the analysis provides a useful foundation, the data and modeling approach require significant refinement. Expanding the data, improving data preprocessing, optimizing model hyperparameters, and enhancing the model itself could all help to maximize its predictive power and generalizability. The current work is a start, but far from the final say on the problem. With time and effort, the approach could be developed into a much more sophisticated, accurate, and robust solution. For now, though, its imperfections and limitations must be acknowledged in interpreting any results or insights it provides.

In summary, the analysis and model show potential, but remain limited and imperfect. With expanded data, enhanced preprocessing, hyperparameter tuning, and model optimization, the approach could be greatly improved. The current model is a start but far from a finished product, and its shortcomings must be kept in mind. Overall, while promising, much work remains to develop a mature, impactful solution.