

PROJECT REPORT

BCIS 5140



11/28/2020

TEAM 6

HAMZA AHMED

LAXMI SUNANDA NAGAREDDYGARI

SAI CHAITANYA BONDADA

SAJA AL OGAIDI

SUHAIL BARI

CONTENTS

* [Introduction](#_INTRODUCTION)
* [Data Sources](#_DATA_SOURCES)
* [ML Problem Formulation](#_ML_PROBLEM_FORMULATION)
* [Data Analysis / Data Cleaning](#_DATA_ANALYSIS_/)
* [Feature Engineering](#_FEATURE_ENGINEERING)
* [Model Training](#_MODEL_TRAINING)
* [Model Evaluation](#_MODEL_EVALUATION)
* [Model Deployment recommendations](#_MODEL_DEPLOYMENT_RECOMMENDATIONS)

# INTRODUCTION

BigMart is a retail store which offers grocery shopping and other day to day shopping products. The data scientists who work for BigMart have collected sales data for the year 2013 which consists of 1559 products across 10 different stores in various cities. They have defined certain attributes of each product which they think could contribute towards whether the product would sell better or not. Our goal is to build a predictive model with high accuracy and find out the sale of each product in various stores depending on various factors which have been incorporated as independent attributes. Using this model, the BigMart Marketing department and executives will try to understand the properties of products which play a key part in increasing sales and hence revenue.

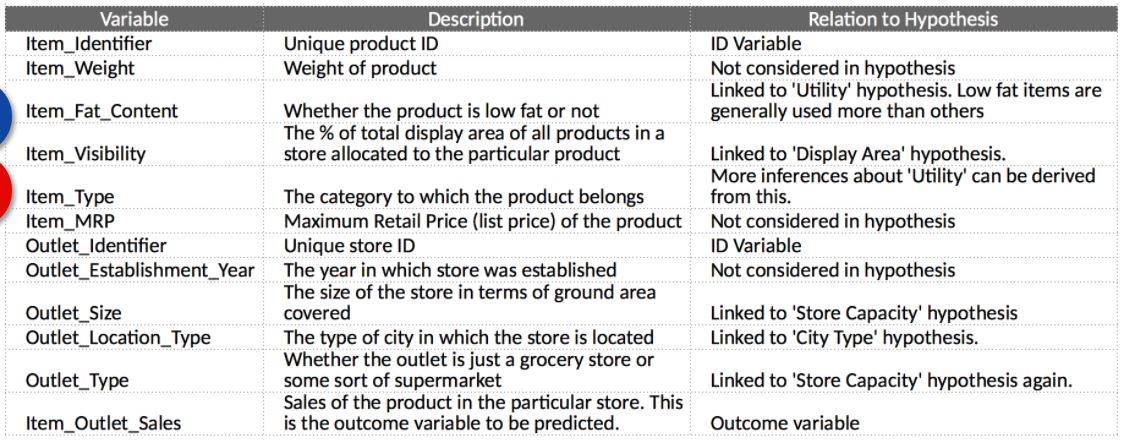
We made some hypotheses which involved a store level hypothesis and a product level hypothesis. Basically, we need to find properties of a product and the store which would impact sales. So for store level hypotheses, City type could be one of the attributes. We can guess that stores located in urban areas would have higher sales because of higher income levels of the people living there. Moreover, store capacity could play a vital role since stores of larger size would have more variety and hence people would prefer going to a one stop shop store. Also, stores having more competitors nearby would have lesser sales. Another metric can be Ambiance - stores which have a humble staff and are better managed would have more customer footprint and hence more sales. Moreover, stores having a good marketing department would know what their customers prefer and will be able to attract customers through their advertising. Other store level metrics can include Location, Customer Behavior and Population Density of the area the store is located in.

Coming to product level hypotheses, metrics like Brand, Packaging, Utility, Display area, Visibility in store, Advertising, Promotional offers and others could affect the sales of products at the stores. Branded products would have higher sales because of higher trust levels from the consumer. Products with better packaging attract customers more. Moreover, day to day products would sell more because they are a necessity as compared to specific use products. Display area matters because products with bigger shelves are likely to catch attention first and sell more. The same way, visibility of the product in the store will matter since the products which are right at the entrance will catch the eye of the customer more than the ones at the back. Similarly, advertising and promotional offers can also dictate which products would sell more depending on what offers are going on for what products.

Therefore, our business problem is to analyze what independent variables matter in predicting our dependent variable which is sales for products in particular stores.

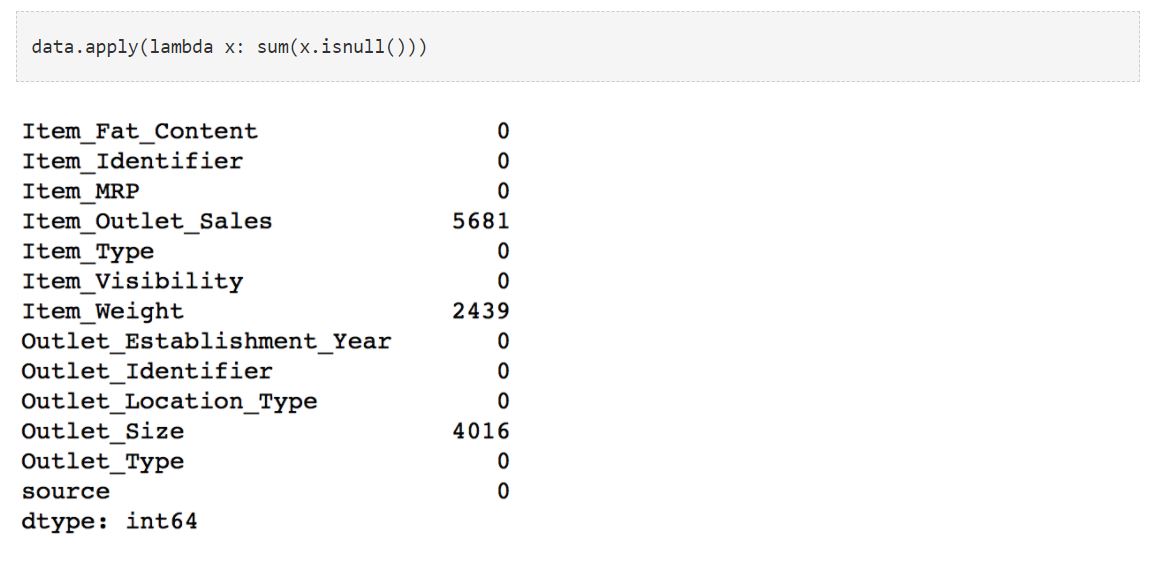
# DATA SOURCES

Our data source was a dataset from a data science competition. It has 12 variables, which are described in the table below:

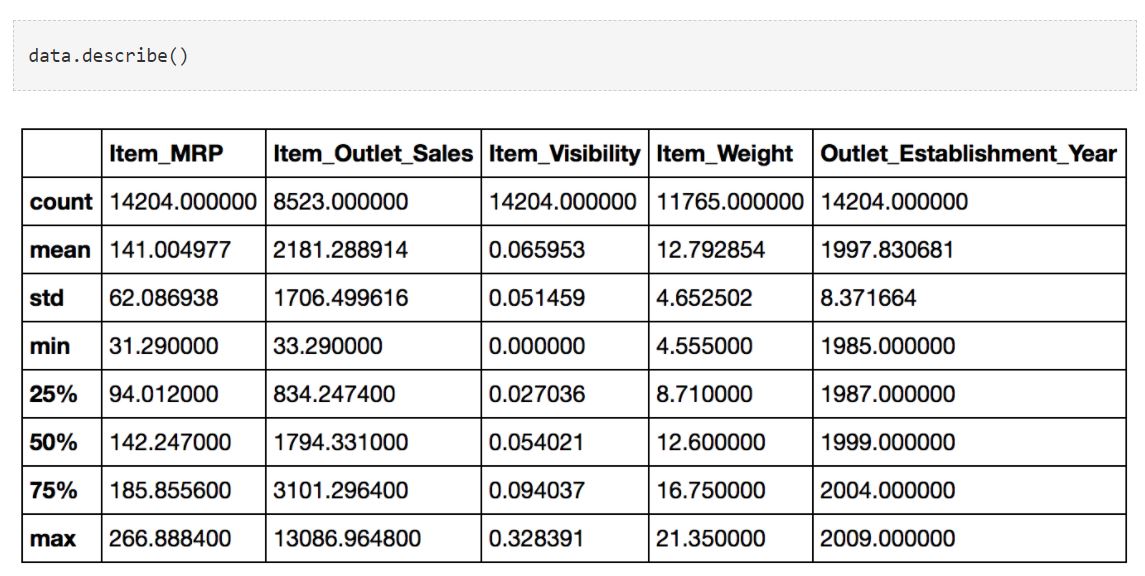


Next, we will figure out irregularities in the data and then perform data cleaning on it.

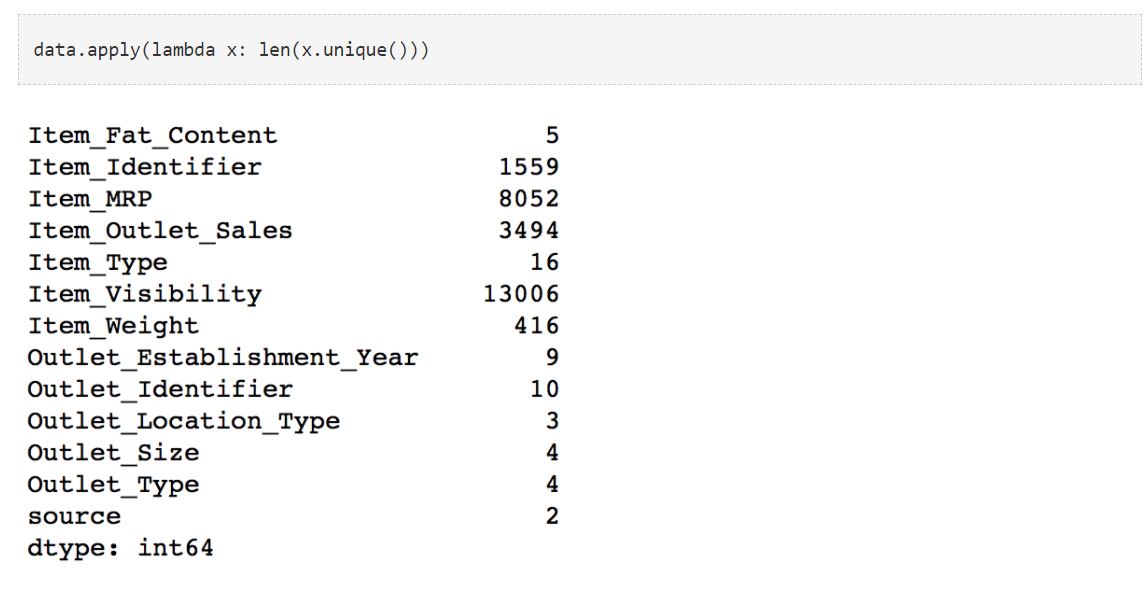
We need to figure out how many null values for each variable are there so we can perform feature engineering on it. Here is how we do it:



Following this, we explore the data to find out statistical metrics for each variable like mean, count, min, max etc:



Finally, we also find out the unique values:

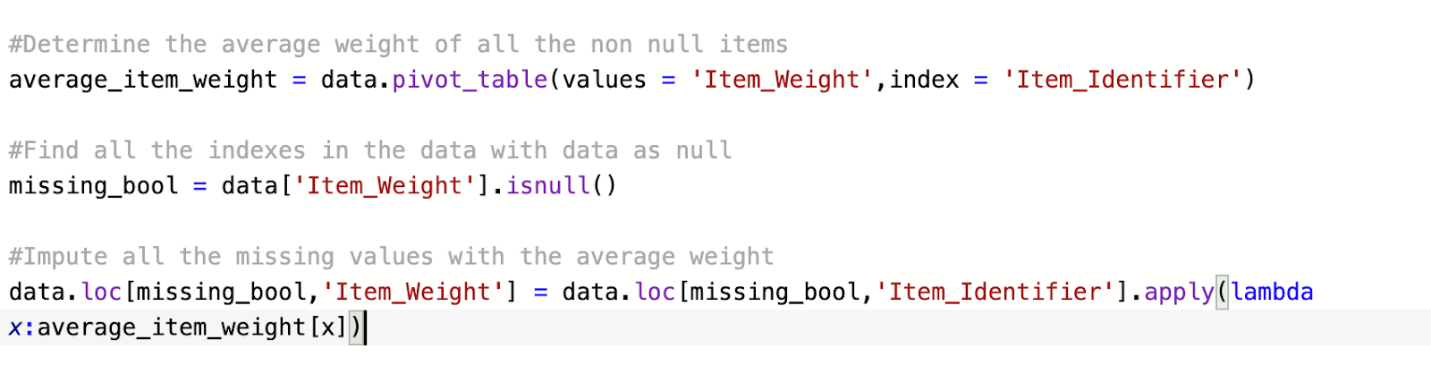


# ML PROBLEM FORMULATION

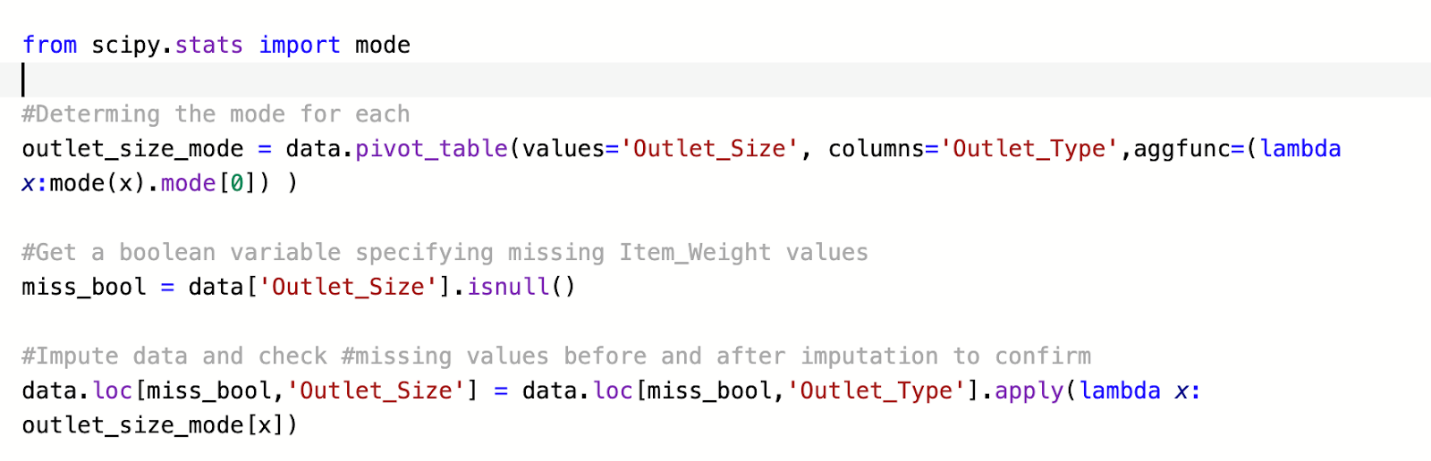
The ML approach used in this project is “Supervised learning”. As per the data, there are few Independent and a dependent attribute which is termed as the “Item\_Outlet\_Sales”. This is the attribute on which the prediction must be made. The prediction technic that has been used is the Regression. Different types of the regressions like Linear, Ridge and the Decision Tree are being used to predict the Sales of a particular item in a particular store. We can use Decision trees both for the sake of Classification and Regression. In this project, it has been utilized for the regression.

# DATA ANALYSIS / DATA CLEANING

The Input data we receive from any data source would generally be incomplete with a lot of the fields being empty and some of them being inconsistent with the reality. This step would generally involve us in dealing with these kinds of data points. We would ideally impute the empty values using the existing data and also try to remove the outlier values from the data source.  
  
In the above data set which we have obtained , we have observed that the variables Item\_Weight and Outlet\_Size are the ones which are missing. For the variable Item\_Weight, we can impute it with the average value of all the available data which contains this variable as not None. This would result in obtaining this column of data to be completely filled.



For the Outlet Size, we can fill them up based on the Outlet\_Mode key and fill it up with respect to its own category mode value

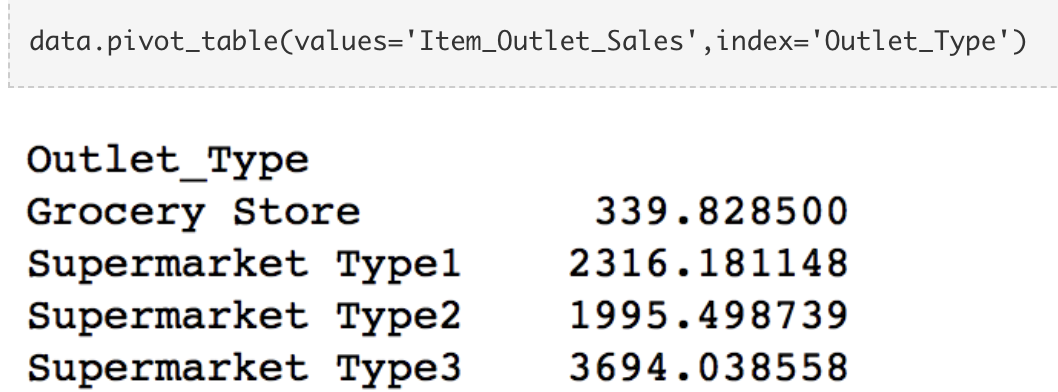


Once we have made sure that there are no missing values in the data, we can go ahead and start working on feature engineering

# FEATURE ENGINEERING

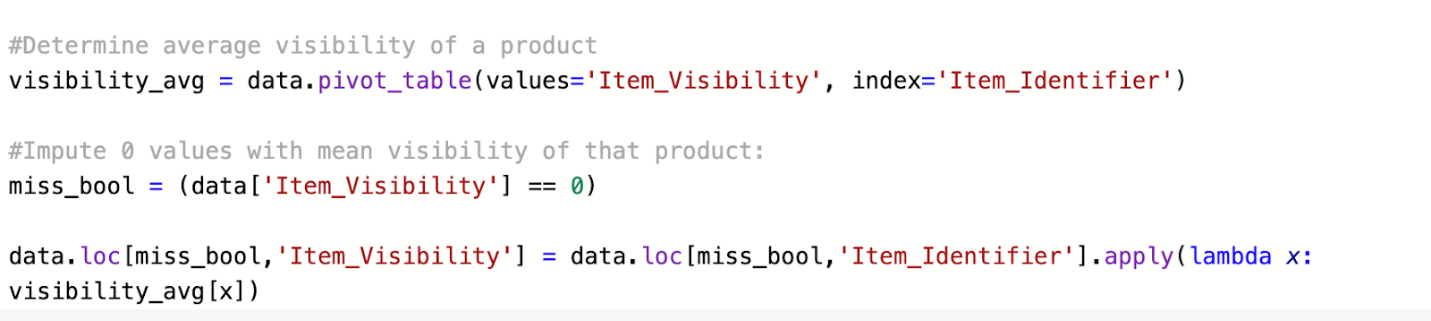
Feature Engineering is the way where with the data set obtained, we try to analyse the existing data and mix few equivalent variables to form custom variables

* We can think about combining Outlet\_size and Outlet\_Sales variables, this can be done if the data supports it.We can try it out by finding the mean of the sales in all the outlet types and look at the results



From the data analysis it's visible that the type and the sales of types do not correlate in a linear way and so we need to keep these variables separate from each other.

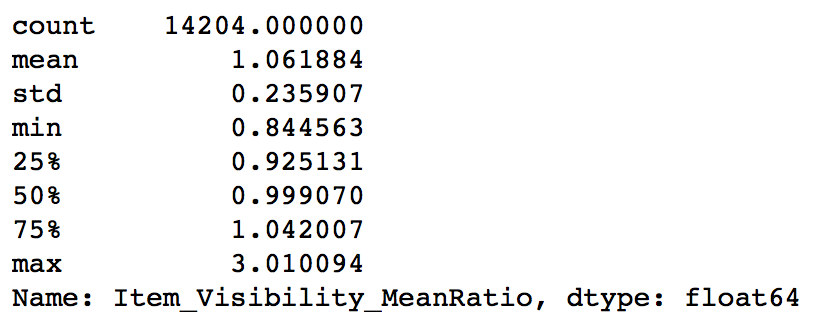
* We noticed that some of the variables like item\_visibility is valued as 0 which doesn't make sense since every item needs to have a non-zero visibility. So here we impute those 0 values with the average visibility of the items of that category.



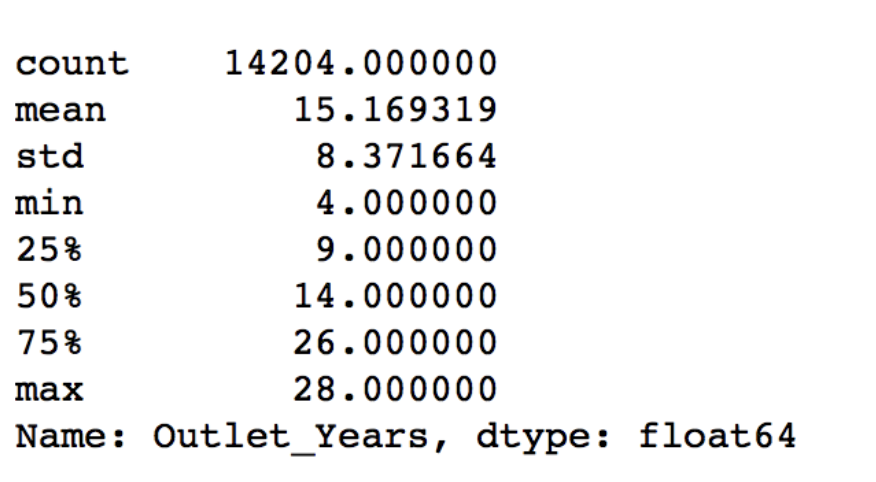
In step 1 we hypothesized that products with higher visibility are likely to sell more. But along with comparing products on absolute terms, we should look at the visibility of the product in that particular store as compared to the mean visibility of that product across all stores. This will give some idea about how much importance was given to that product in a store as compared to other stores. We can use the ‘visibility\_avg’ variable made above to achieve this.

#Determine another variable with means ratio

data['Item\_Visibility\_MeanRatio'] = data.apply(lambda x: x['Item\_Visibility']/visibility\_avg[x['Item\_Identifier']], axis=1)



* We can obtain the years of establishment by subtracting the year established with the current year

data['Outlet\_Years'] = 2020 - data['Outlet\_Establishment\_Year']

* Modifying and Normalizing the variable Item\_Fat\_Content as we have found the same variable type in different formats.

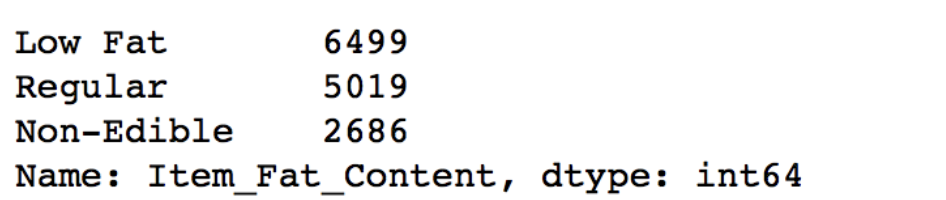
data['Item\_Fat\_Content'] = data['Item\_Fat\_Content'].replace({'LF':'Low Fat',

                                                             'reg':'Regular',

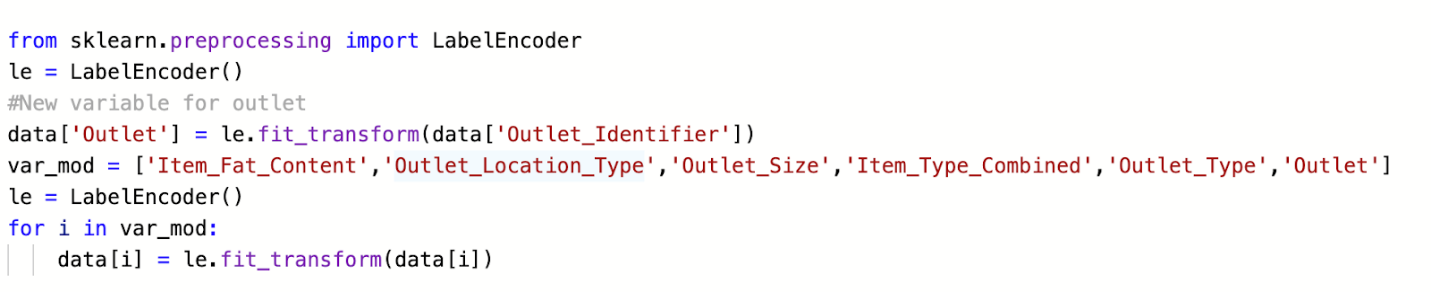
                                                             'low fat':'Low Fat'})

There are non-consumable items and they should not be having any type of fat content variable, So we need to create a separate category for those kind of items

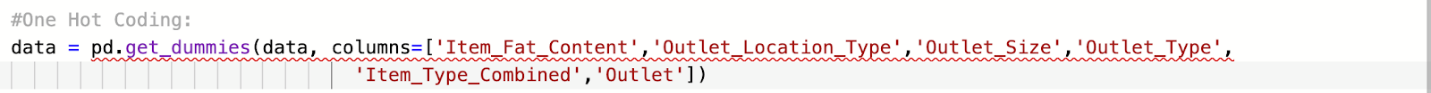
data.loc[data['Item\_Type\_Combined']=="Non-Consumable",'Item\_Fat\_Content'] = "Non-Edible"



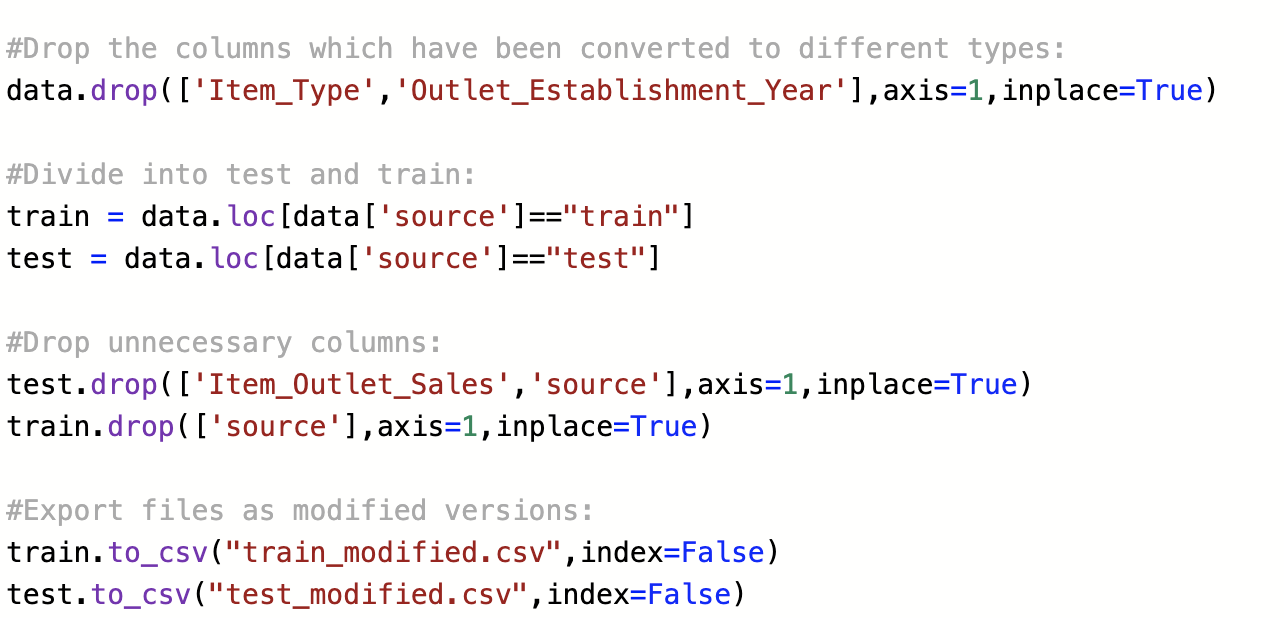
* The variables present in the data are of multiple data types, for example outlet\_identifier and these variables need to be converted into numerical types because scikit learn only works with numerical variables. We can start coding all categorical variables as numeric variables using the LabelEncoder



One Hot Encoding is another way to convert variables of non numeric types to numeric types by assigning binary values to different categories.For example, the Item\_Fat\_Content has 3 categories – ‘Low Fat’, ‘Regular’ and ‘Non-Edible’. One hot coding will remove this variable and generate 3 new variables. Each will have binary numbers – 0 (if the category is not present) and 1(if category is present). This can be done using the ‘get\_dummies’ function of Pandas.All variables are now float and each category has a new variable.



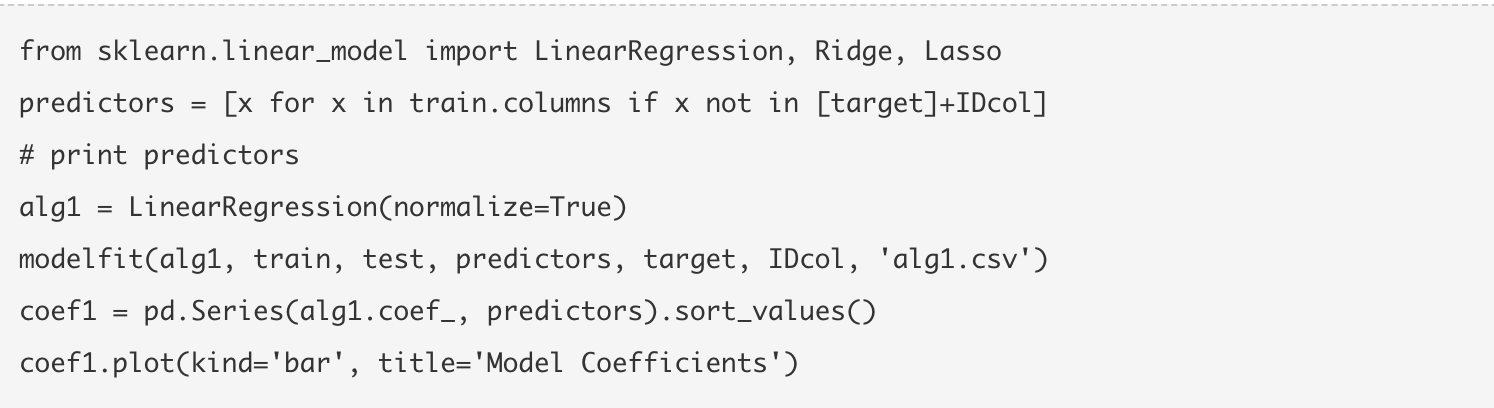
* After all the data transformation is done we need to convert the data back into train test data



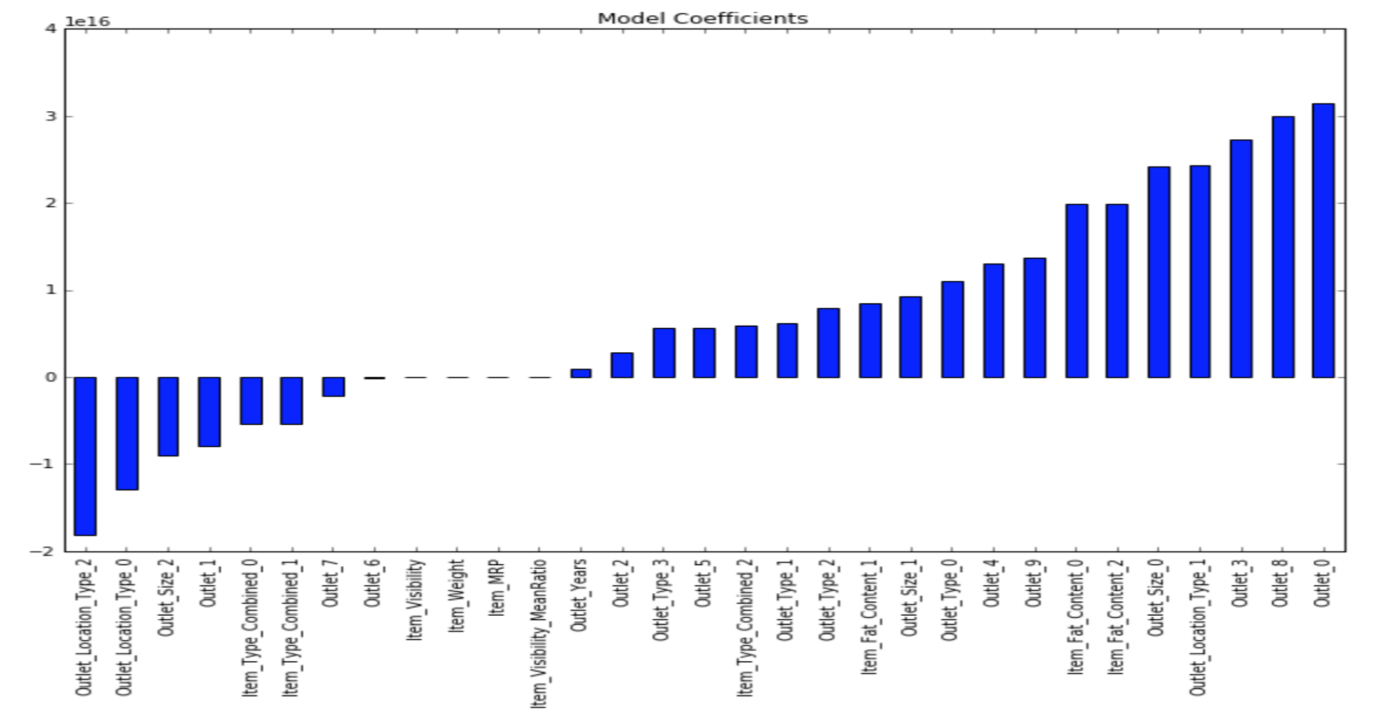
# MODEL TRAINING

Now that we have all the data prepared, we need to build models on top of this data. First thing which can be done is building a baseline model which requires making no predictive model but just making an informed guess regarding the output. For the baseline prediction we can get a rough estimate on the predicted values.

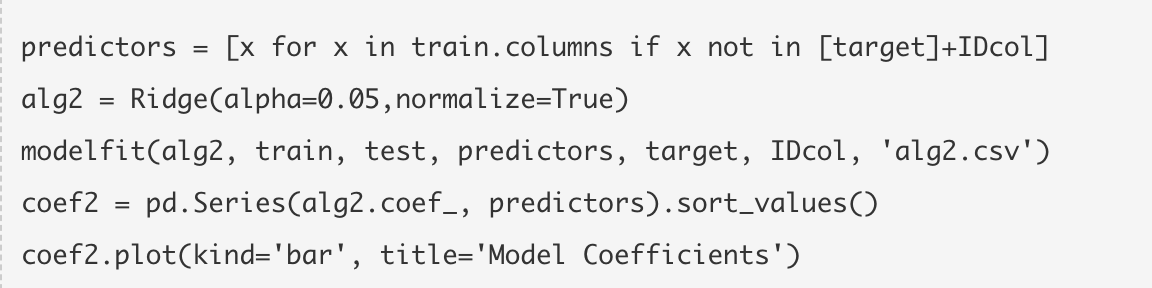
The first model which we can try on the data set would be linear regression

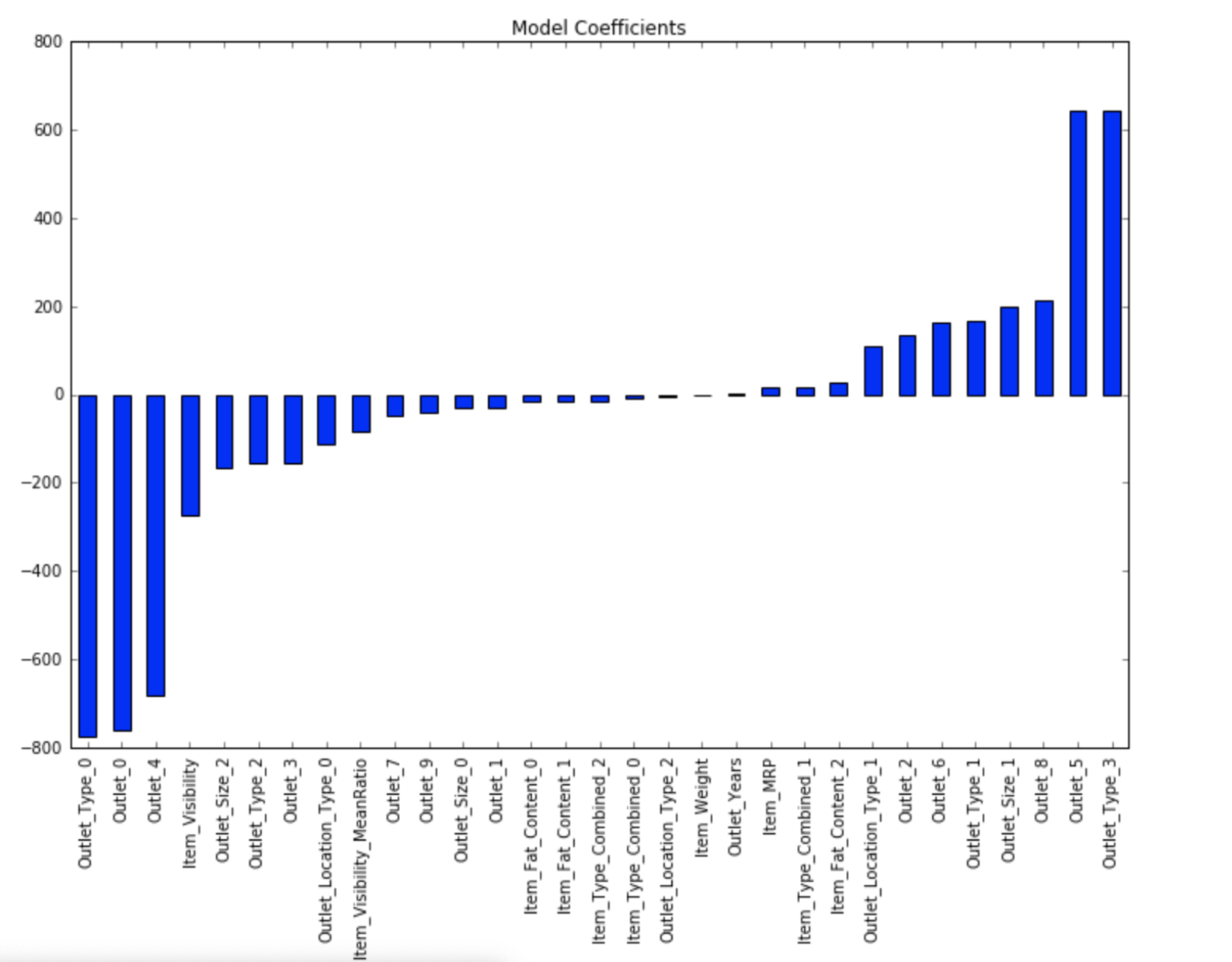


We can model the coefficients of the resulting linear regression models and with large values are coefficients, it could be a case of overfitting, but this model would definitely be better than the baseline prediction.



* To overcome the case of large coefficients we can use the ridge regression model using Lasso Technique.

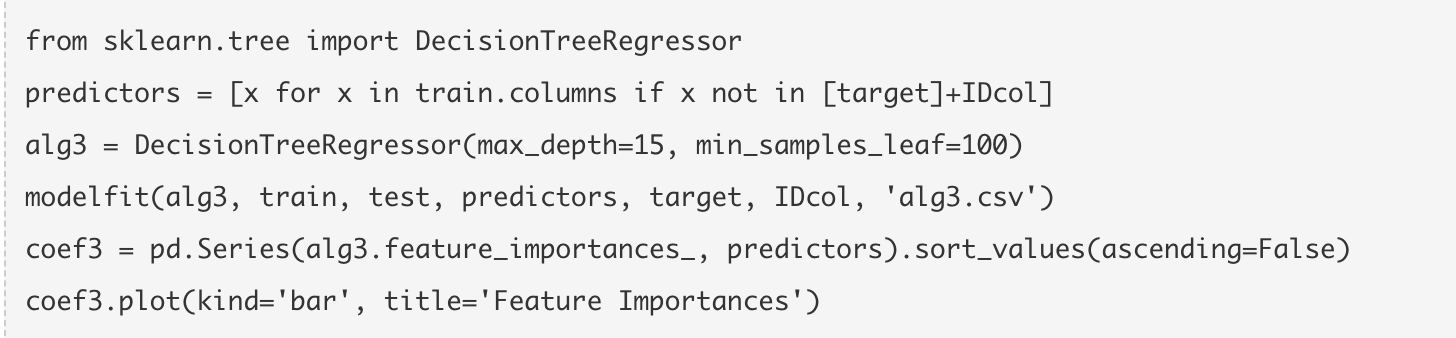


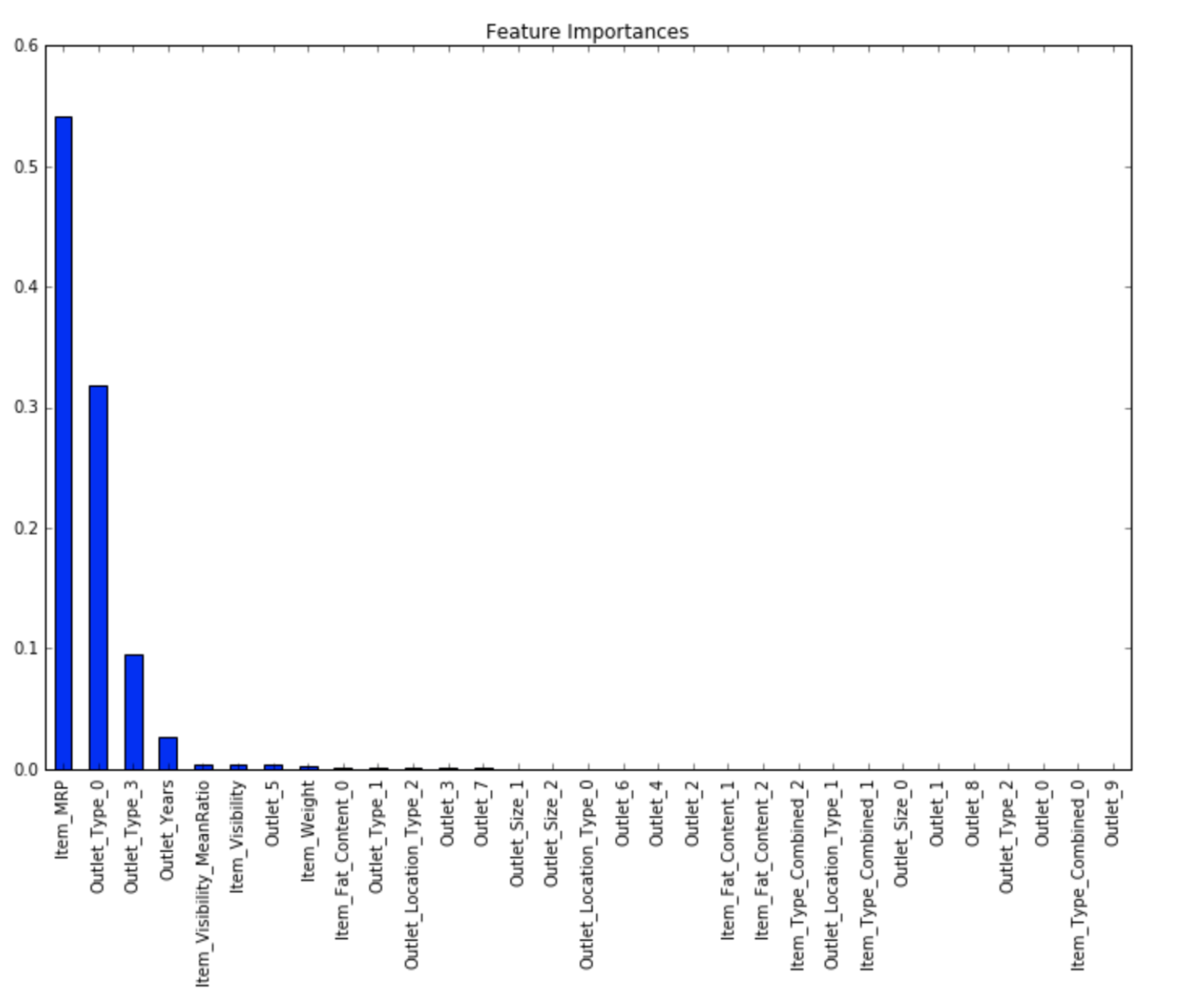


The coefficients seem better here compared to the linear regression case but there doesn't seem to be much change in the validation score from this model and would likely be the case even with fine tuning the parameter

* Decision Tree Model

We can try out the decision tree model with 15 layers and 100 nodes





Here you can see that the RMSE is 1058 and the mean CV error is 1091. This tells us that the model is slightly overfitting. We can achieve a slight improvement by making a decision tree with just top 4 variables, a max\_depth of 8 and min\_samples\_leaf as 150.

# MODEL EVALUATION

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

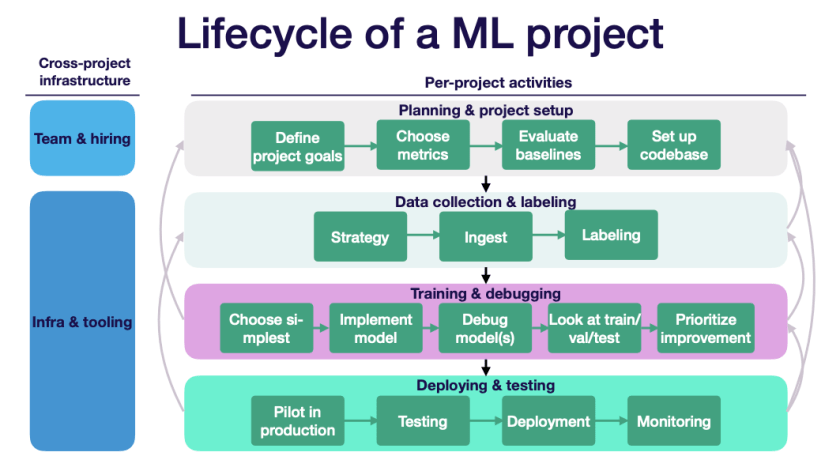
Graphical user interface, text

Description automatically generated

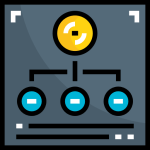
The results of the cross validations are displayed below for each of the three algorithm code snippets. The error should least and the R-Square should be highest, The Decision Tree has the least Root Mean Square error and the R^2 value is the highest. So, the Decision Tree is the best model to predict the Item\_Sales .

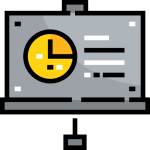
# MODEL DEPLOYMENT RECOMMENDATIONS

**In any project, before we dive into the schematics of what tools are to be used, it is always recommended to go over the basic ideas and implementation of the project from every single aspect.**  To attain that understanding, it’s helpful to put yourself in the shoes of a software engineer. Deployment is entirely distinct from routine machine learning tasks like feature engineering, model selection, or model evaluation. So, let’s go over how we decide to deploy a model.



**Deploying a Lead Scoring Model**

Suppose a data scientist has built a lead scoring model for a group of technical analysts who are well versed in SQL. The analysts seek to group new leads into buckets based on their likelihoods of converting into customers.



Each morning they would like to use data from the database to create/update dashboards they maintain in a BI tool.

Since the analysts know SQL and expect model scores to be stored in the database, "deploying" the lead scoring model means generating daily lead scores for new leads and storing these in the analysts’ database.

The key aspects of this deployment are:

1. Predictions can be generated on a group of new leads,
2. These predictions need to be made available each day, and
3. The predictions need to be stored in a database. The deployment process needs to satisfy these three constraints in order for the ML model to add value to the business.