# Mini-project report 1 – predicting future sales at a supermarket

By: Martin Haver

## 1. Problem Statement:

BigMart (a supermarket chain) collected 2013 sales data for 1559 products across 10 stores in different cities in the USA. 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 wants to try to understand the properties of products and stores which play a key role in increasing sales.

In another words, the goal is to find out which properties of a product influences its sales.

## 2. The data

The data represent 2013 sales for 1559 products across 10 stores in different cities. They came split into train.csv (8524 rows) and test.csv (5682 rows). After initial data analysis, these are the results describing the dataset:

**Frequency of Categories for varible Item_Fat_Content**

Low Fat 8485

Regular 4824

LF 522

reg 195

low fat 178

Name: Item_Fat_Content, dtype: int64

**Frequency of Categories for varible Item_Type**

Fruits and Vegetables 2013

Snack Foods 1989

Household 1548

Frozen Foods 1426

Dairy 1136

Baking Goods 1086

Canned 1084

Health and Hygiene 858

Meat 736

Soft Drinks 726

Breads 416

Hard Drinks 362

Others 280

Starchy Foods 269

Breakfast 186

Seafood 89

Name: Item_Type, dtype: int64

**Frequency of Categories for varible Outlet_Location_Type**

Tier 3 5583

Tier 2 4641

Tier 1 3980

Name: Outlet_Location_Type, dtype: int64

**Frequency of Categories for varible Outlet_Size**

Medium 4655

Small 3980

High 1553

Name: Outlet_Size, dtype: int64

**Frequency of Categories for varible Outlet_Type**

Supermarket Type1 9294

Grocery Store 1805

Supermarket Type3 1559

Supermarket Type2 1546

Name: Outlet_Type, dtype: int64

## 3. Data cleaning and feature engineering

Test and train data were merged to avoid performing data cleaning twice. After a quick analysis, it was revealed that there are missing values in “Item weight” column. Missing weight values were substituted by average weight of items in the dataset.

I thought that maybe the age of the store may be important so I created a  value reflecting this, bearing in mind that the data come from year 2013.

Some items had Visibility (exposure percentage in the supermarket) set to zero, which I considered impossible, therefore these zero values were replaced by set´s visibility mean value

There are 16 different categories of item, but they are really only of 3 types, as suggested by their IDs. These are Food, Drinks and Non-consumables. Therefore I created these three new categories in the dataset for better analysis.

Typos in category Item_fat were corrected, for example “reg” was corrected to “Regular”

Some non-consumables had fat value specified for them which seems incorrect, therefore they were moved into a specific category “non-edibles”

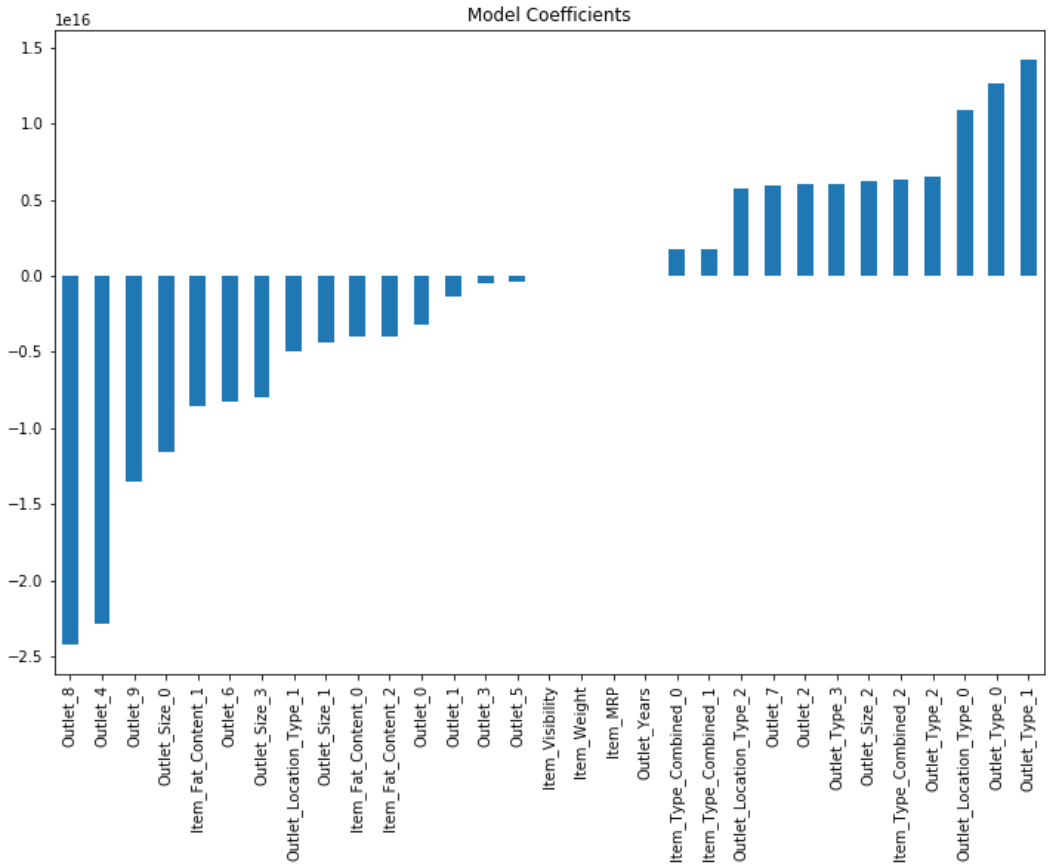
One-hot coding was performed to comply with scikit-learn demand for only numeric variables. This means introducing dummy variables instead of nominal ones.

## 4. Model building

I experimented with two types of models (Linear regression and Random forest) with these results:

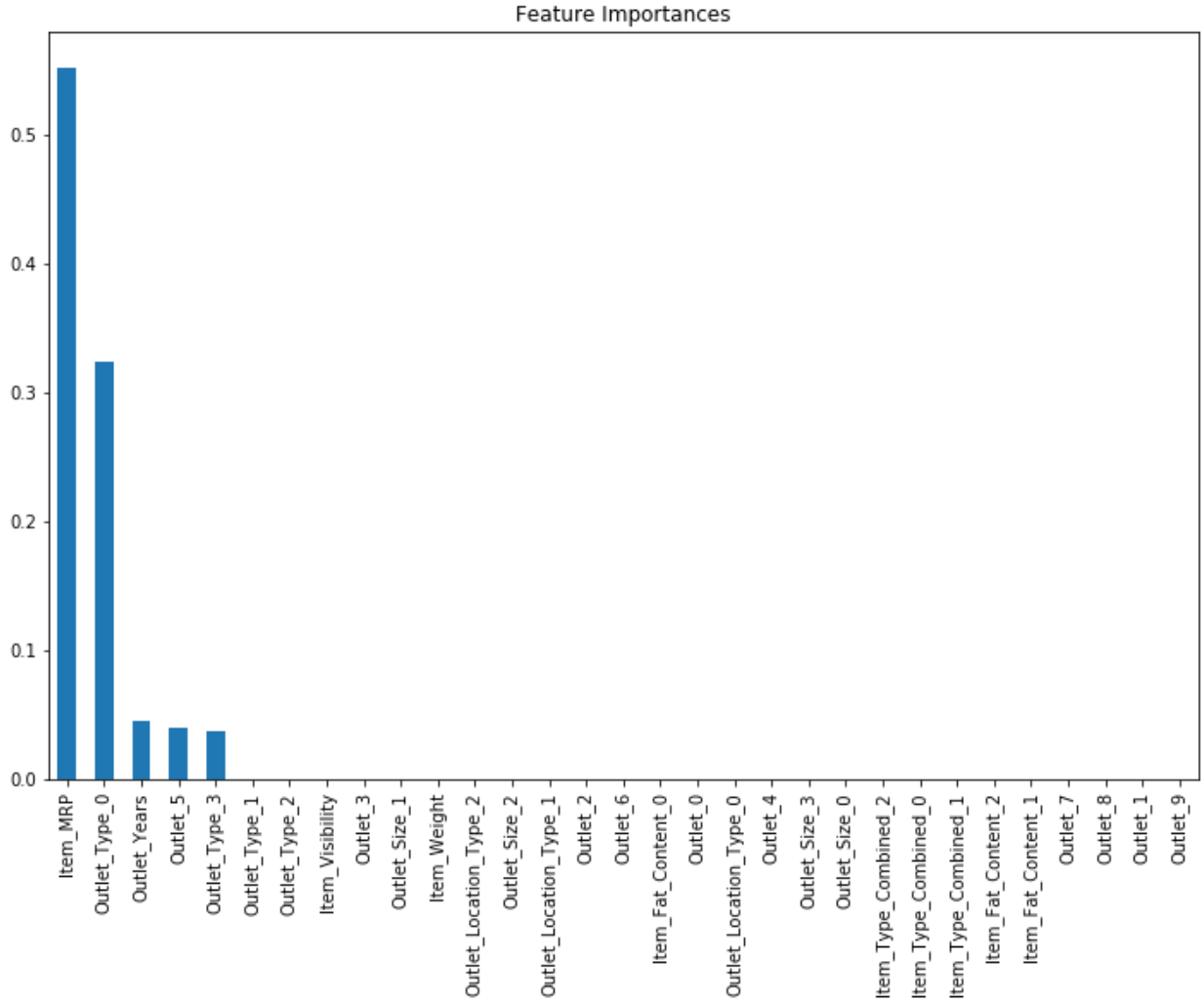
**Linear regression:**

```
Model Report
RMSE: 1128
CV Score: Mean - 1129 | Std - 43.25 | Min - 1075 | Max - 1209
```



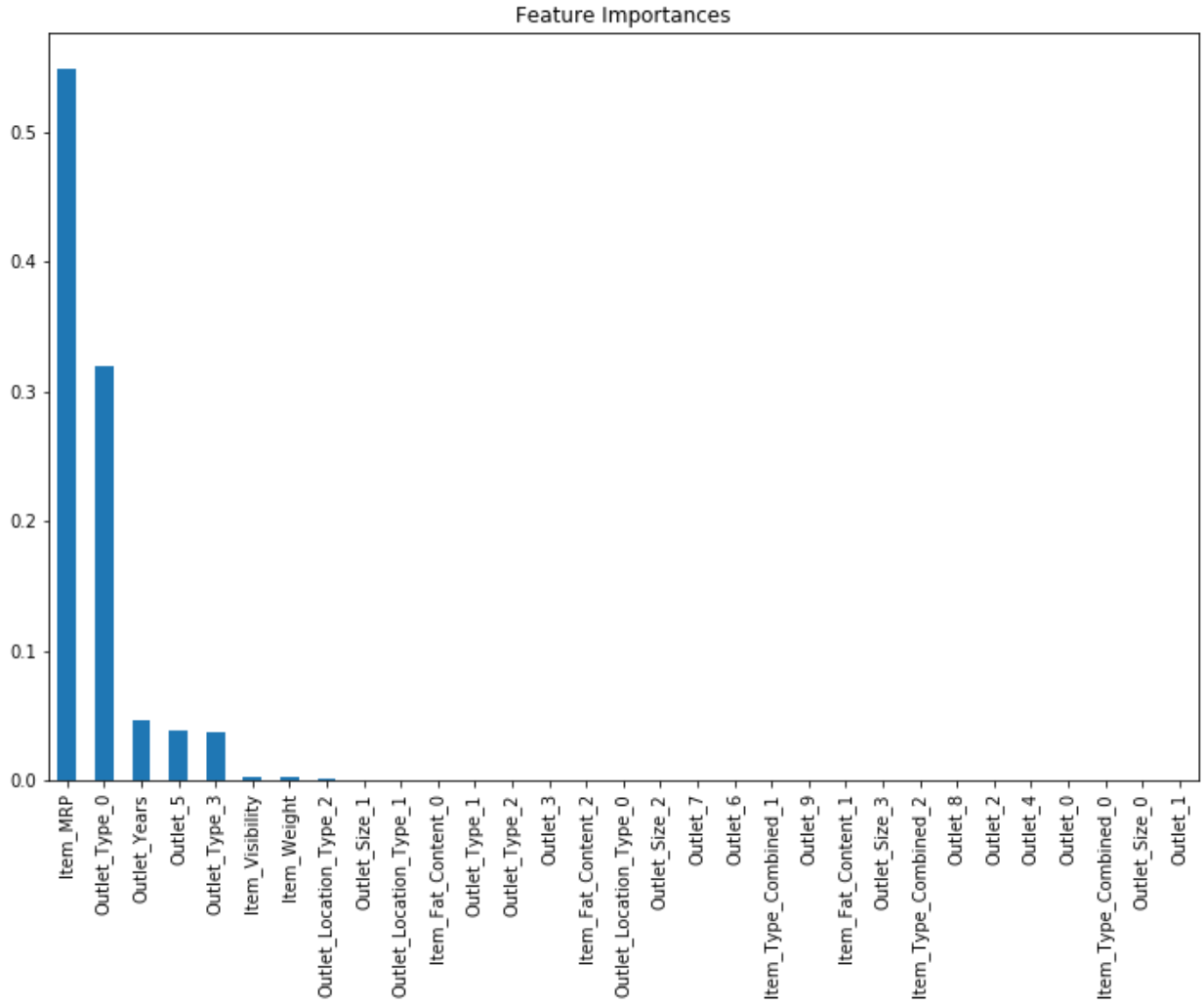
**Random forest with depth of 5 and 200 trees:**

```
Model Report
RMSE: 1073
CV Score: Mean - 1083 | Std - 43.97 | Min - 1018 | Max - 1160
```




**Random forest slightly tuned with max_depth of 7 and 500 trees.**

```
Model Report
RMSE: 1064
CV Score: Mean - 1083 | Std - 42.91 | Min - 1022 | Max - 1159
```




## 5. Conclusion

In this mini project I practiced how to clean the dataset, prepare features for machine learning purposes. Then I experimented with linear regression and random forest algorithms to predict future sales of a product. The results, measured by CV (cross-validation) error are better for Random forest model. I tried to tune the algorithm, increasing the depth and number of trees to make the model more robust but this did not improve the error and proved to be more computationally expensive. The language used was Python with Pandas and Sklearn libraries.