# Big mart Sales Prediction using classification

## 1. Introduction

### 1.1 Project Motivation

For any supermarket its revenue matters a lot. So, to analyse revenue it is necessary to know the product sale. So, by focusing on this matter we have decided to build a predictive model and find out the sales of each product at a particular BigMart store. Using this model, BigMart will try to understand the properties of products and stores which play a key role in increasing sales.

### 1.2 Aims and Objectives

The aim is to build a predictive model and find out the sales of each product at a particular store using Apache Mahout on the top of Hadoop map reduce architecture and Machine Learning algorithms such as Classification and Regression.

### 1.3 Report Structure

- Background/History of the Study
- Approach and Implementation
- Experiment Result and Discussions

## 2. Background/History of the Study

Retail is the industry, which extensively uses analytics to optimize business processes whether it is online retail or offline retail. Retailers want to maximize their sales and profit. Tasks like product placement, inventory management, customized offers, product bundling etc. are being smartly handled using data science techniques. In addition to this, Hadoop technology provides the benefits such as flexibility, cost effectiveness, scalability, faster throughput, protection against failure etc. So, ultimately the main objective is to use widely growing technology such as Hadoop and Machine Learning algorithms to predict the sales of a Big Mart and make a little contribution to the Retail Industry.

# 3. Approach and Implementation

- **Data Set**:
  We have train (8523) and test (5681) data set, train data set has both input and output variable(s).

| **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 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 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 |
| **Item_Outlet_Sales** | Sales of the product in the particulate store. This is the outcome variable to be predicted. |

- **Data Pre-processing**

1. <u>Hypothesis Generation</u>

   - There are some store level Hypothesis such as City Type, Population, Location, Competitors, Store Capacity etc. and some Product level Hypothesis such as Brand, Visibility in Store, Packaging, Utility, Offers etc. that can impact the outcome.

2. <u>Data Exploration</u>

   - Check how many columns have missing values.
   - Check how many unique values in each column.
   - Filtering of categorical variables.

3. <u>Data Cleaning</u>

   - Imputing missing by calculating average or mode values

4. <u>Feature Engineering</u>

- Created new variables to improve model performance.
- Modified categories of some variables.

```
Frequency Table

Frequency table for varible Item_Fat_Content
Low Fat     8485
Regular     4824
LF           522
reg          195
low fat      178
Name: Item_Fat_Content, dtype: int64

Frequency table 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 table for varible Outlet_Location_Type
Tier 3    5583
Tier 2    4641
Tier 1    3980
Name: Outlet_Location_Type, dtype: int64
```

```
Frequency table for varible Outlet_Size
Medium    4655
NaN       4016
Small     3980
High      1553
Name: Outlet_Size, dtype: int64

Frequency table for varible Outlet_Type
Supermarket Type1    9294
Grocery Store        1805
Supermarket Type3    1559
Supermarket Type2    1546
Name: Outlet_Type, dtype: int64
Orignal #missing: 2439
Final # missing: 0
```

```
Mode for each Outlet_Type:
Outlet_Type Grocery Store Supermarket Type1 Supermarket Type2  \
Outlet_Size         Small             Small            Medium

Outlet_Type Supermarket Type3
Outlet_Size            Medium

Orignal #missing: 4016
Final # missing: 0
Number of 0 values initially: 879
Number of 0 values after modification: 0
count    14204.000000
mean         1.061884
std          0.235907
min          0.844563
25%          0.925131
50%          0.999070
75%          1.042007
max          3.010094
Name: Item_Visibility_MeanRatio, dtype: float64
```

## Modified File after pre-processing

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FDA15 | 249.8092 | 3735.138 | 0.016047 | 9.3 | OUT049 | 0.931078 | 14 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| DRC01 | 48.2692 | 443.4228 | 0.019278 | 5.92 | OUT018 | 0.93342 | 4 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| FDN15 | 141.618 | 2097.27 | 0.01676 | 17.5 | OUT049 | 0.960069 | 14 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| FDX07 | 182.095 | 732.38 | 0.017834 | 19.2 | OUT010 | 1 | 15 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| NCD19 | 53.8614 | 994.7052 | 0.00978 | 8.93 | OUT013 | 1 | 26 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| FDP36 | 51.4008 | 556.6088 | 0.057059 | 10.395 | OUT018 | 1 | 4 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| FDO10 | 57.6588 | 343.5528 | 0.012741 | 13.65 | OUT013 | 1.497197 | 26 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| FDP10 | 107.7622 | 4022.764 | 0.12747 | 19 | OUT027 | 0.870493 | 28 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| FDH17 | 96.9726 | 1076.599 | 0.016687 | 16.2 | OUT045 | 0.92416 | 11 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| FDU28 | 187.8214 | 4710.535 | 0.09445 | 19.2 | OUT017 | 0.963983 | 6 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| FDY07 | 45.5402 | 1516.027 | 0.040627 | 11.8 | OUT049 | 1 | 14 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| FDA03 | 144.1102 | 2187.153 | 0.045464 | 18.5 | OUT046 | 1.036695 | 16 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| FDX32 | 145.4786 | 1589.265 | 0.100014 | 15.1 | OUT049 | 1.02636 | 14 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| FDS46 | 119.6782 | 2145.208 | 0.047257 | 17.6 | OUT046 | 0.92229 | 16 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| FDF32 | 196.4426 | 1977.426 | 0.068024 | 16.35 | OUT013 | 1.171331 | 26 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

## Description of Decision Tree

- The first step is to generate a decision tree using training dataset. Core algorithm to build decision tree are called ID3 which employs top down approach. To construct decision tree for regression Standard Deviation Reduction will be used.
- Step 1: The standard deviation of the target is calculated.

$$S = \sqrt{\frac{\sum (x-\mu)^2}{n}}$$

- *Step 2*: The dataset is then split on the different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.

$$SDR(T,X) = S(T) - S(T,X)$$

- *Step 3*: The attribute with the largest standard deviation reduction is chosen for the decision node.
- *Step 4a*: Dataset is divided based on the values of the selected attribute.
- *Step 4b*: A branch set with standard deviation more than 0 needs further splitting.
- *Step 5*: The process is run recursively on the non-leaf branches, until all data is processed.
- When the number of instances is more than one at a leaf node we calculate the *average* as the final value for the target.
- Now tree is generated so we can pass the test data to the tree and can predict the value.

**Description of Random Forest**

- For tree b= 1 to B; (B is the number of trees):
    - Draw a bootstrap sample from the training data
    - Select m features at random from the total p features.
    - Grow a ID3 tree $T_b$ to the bootstrap data
- Output the ensemble of trees
- Pass the test data to each tree and collect their output classes
- Choose a final class by averaging the classes from the above list of classes

**Model Creation**

- Using python machine learning library scikit-learn, we created models for both Decision tree and Random forest algorithms.
- We have done cross-validation and got root mean squared error and cross validation score as mean, min, max and standard deviation to check accuracy of our model.
  Then we passed testing data on created model and got prediction for value of sales

**Environmental Setup for Apache Mahout Map Reduce**

- VirtualBox-5.2.0
- Linux linuxmint-18.2-cinnamon-64bit
- Java SE 7 (jdk1.7.0_80)
- Hadoop Setup for Single Node Cluster
- Apache Maven 3.5.2
- Apache Mahout 0.11.0

**Implementation of Classification**

- Put Dataset into Hadoop File System.
- Prepare the description file that describe type of variables using mahout-core-0.5-job.jar with following command:

```
mahout describe –p /input_data/train.csv –f /input_data/in.info –d I 4 N C 32 N L
```

- Split data into Train and Test set by specifying percentage for each of them.

```
mahout splitDataset --input /input_data/train.csv –output /output_data --trainingPercentage 0.7 --probePercentage 0.3
```

- Build model using mahout-examples-0.11.0-job.jar which has the implementation of Machine Learning algorithms in mapreduce for Trainset.

  mahout buildforest –d /output_data/trainingSet/* -ds /input_data/in.info –sl 3 –p –t 10 –o /output_model

- We can build a Decision Tree by using following command, which similar to above command but need to update two parameters.

  mahout buildforest –d /output_data/trainingSet/* -ds /input_data/in.info –sl 39 –p –t 1 –o /output_model

- Test Model for Test set.

  mahout testforest –i /output_data/probeSet –ds /input_data/in.info –m /output_model –a –mr –o /output_prediction

# 4. Experiment Results and Discussion

- Classification and Regression both are widely used techniques in Data Science. As an individual team we have implemented classification algorithm. But, we were curious to know which technique works better in terms of accuracy and runtime. So, we congregated with another team who were implementing Regression technique using similar dataset to examine aspects of both algorithms.

**Experiment Results using Non- Parallel Approach**

1) <u>Regression</u>:-

- Linear Regression

```
Model Report
RMSE : 1076
CV Score : Mean - 1078 | Std - 41.03 | Min - 1022 | Max - 1166
```

- Ridge Regression

```
Model Report
RMSE : 1076
CV Score : Mean - 1078 | Std - 41.79 | Min - 1022 | Max - 1168
```

2) <u>Classification</u>:-

- Decision Tree

```
Model Report
RMSE : 1001
CV Score : Mean - 1030 | Std - 40.27 | Min - 963.4 | Max - 1113
```

- Random Forest

```
Model Report
RMSE : 1027
CV Score : Mean - 1033 | Std - 39.32 | Min - 960.2 | Max - 1122
```

## ➢ Experiment Results using Parallel Approach

- Random Forest

```
Statistics
-------------------------------------------------------
Kappa                                      0.0175
Accuracy                                  63.4766%
Reliability                               41.6085%
Reliability (standard deviation)           0.1127
17/12/03 19:55:23 INFO MahoutDriver: Program took 198699 ms (Minutes: 3.31165)
```

- Regression

```
AUC = 0.50
confusion: [[0.0, 0.0], [2.0, 8521.0]]
entropy: [[NaN, NaN], [0.0, 0.0]]
17/12/03 19:25:23 INFO MahoutDriver: Program took 6323 ms (Minutes: 0.10538333333333333)
```

# 5. Conclusion

- From both parallel and non-parallel approaches we conclude that
- Classification works better for this dataset than Regression in terms of efficiency.
- But regression works faster than Classification.

# 6. References


1. https://www.analyticsvidhya.com/blog/2016/02/bigmart-sales-solution-top-20/
2. https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/
3. https://www.youtube.com/watch?v=pPP39NTjZR8-Classification
4. https://www.youtube.com/watch?v=mJW7na1HlAQ- Regression
5. https://www.youtube.com/watch?v=FEHh1GVg3ww- Hadoop