**Data Analysis of Sales Prediction of Big Mart Data**



A project paper is submitted for the degree of B.Sc. Examination 2018 in Computer Science and Engineering

**Project supervisor**

Dr. Shamim Ahmad

Professor

Dept. of Computer Science and Engineering,

University of Rajshahi

**Submitted BY**

Name: Md. Ismail Hossain

ID: 1510276139

Session: 2014-15

Exam: B.Sc. part-4 even semester

Dept. of Computer Science and Engineering,

University of Rajshahi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY OF RAJSHAHI

***Dedicated***

***To***

***My Parents &***

***Teachers***

**ACKNOWLEDGEMENT**

Thanks to Almighty Allah, the creator and sustainer, who have given me strength, patience and opportunity to complete the project, ‘Data analysis of Big Mart sales data’.

I wish to express deep gratitude and respect to my project supervisor **Dr. Shamim Ahmad, Professor, Dept. of Computer Science and Engineering, University of Rajshahi** for the suggestion of the topic, dissertation and constant guidance, advice encouragement and every possible help throughout the work which paved the completion of this project

I wish to thank all of my respectable teachers of the department of Computer Science & Engineering, University of Rajshahi for their valuable suggestion and academic help during the course of work.

I am grateful to all my friends who encourage and help to solve in various critical section for during this project. I am earnestly indebted to my parents for their encouragement and love that all this worthwhile.

**ABSTRACT**

This work is concerned with the coding and analysis of “Big **Mart Data**” which contains Hypothesis generation, Data exploration, Data cleaning, Feature engineering, Model building. We 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*.*

**CONTENT**

**Chapter One: Introduction ………………………………… 10-13**

1.1 Introduction ………………………………………………………. 11

1.2 Why this project is being chosen ……………………….. 11

* 1. Objective …………………………………………………………. 12

1.4Organization of the project ………………………………… 12

1.5 Conclusion ………………………………………………………….. 12

**Chapter Two: Introduction ………………………………… 14-29**

2.1 Introduction ………………………………………………………. 15

2.2 Data analysis ………………………………………………………. 15

**Chapter Three: Tools ………………………………………… 20-30**

3.1 Software needed to run this project ……………… 21

3.2 Data analysis languages …………………………………. 21

3.3 Why Python …………………………………………………… 25

3.4 Numpy …………………………………………………………… 26

3.5 Pandas …………………………………………………………… 27

3.6 Anaconda ………………………………………………………. 29

3.7 Spyder ……………………………………………………………. 30

**Chapter Four: Coding and Analysis…………… 33-47**

4.1 Stages …………………………………………………………… 34

4.2 Hypothesis Generation …………………………………. 34

4.3 Data Exploration …………………………………………… 36

4.4 Data Cleaning ………………………………………………… 38

4.5 Feature Engineering ……………………………………… 49

4.6 Model Building ……………………………………………… 45

4.6.1 Mean based Model ……………………………. 45

4.6.2 Linear Regression Model ……………………. 45

4.6.3 Ridge Regression Model …………………….. 47

**Chapter Five: Discussion……………………………… 50-55**

5.1 Output Discussion ………………………………………… 51

**Chapter Six: Conclusion ……………………………… 56-57**

6.1 Conclusion …………………………………………. 57

**CHAPTER ONE**

**INTRODUCTION**

**1.1 Introduction**

The objective of this project is to analyze sales information of some popular e-commerce sites like amazon, e-bay, Alibaba, Coursera and Udacity. This analysis will produce some report on their sales’ pattern, relation between sales’ rate and price of the product etc. This type report may help to make a good business planning. Here I have chosen BigMart data for analysis.

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

* 1. **Why this project is being chosen**

This project will analyze the BigMart sales data and help to take better decision for future. This project will help to make better sales plan for future

* 1. **Objective**

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.

* 1. **Organization of the project**

This project is organized in details in this project paper. The hypothesis generation and coding issue is illustrated in chapter 2. This chapter describe about the python 3.5, numpy and panda (Data centric python packages). In chapter 3 elaboration of analysis

1. Hypothesis generation
2. Data exploring
3. Data cleaning
4. Feature engineering
5. Model building
   1. **Conclusion**

The data analysis of online business site, social media etc. are drawing much attention of data scientists in recent days.

So it’s the time to learn data analysis for making better decision about the future.

**CHAPTER TWO**

**BACKGROUND STUDY**

**2.1 Introduction**

Data analysis is a process of inspecting, [cleansing](https://en.wikipedia.org/wiki/Data_cleansing), [transforming](https://en.wikipedia.org/wiki/Data_transformation), and [modeling](https://en.wikipedia.org/wiki/Data_modeling) [data](https://en.wikipedia.org/wiki/Data) with the goal of discovering useful information, informing conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, while being used in different business, science, and social science domains. In today's business, data analysis is playing a role in making decisions more scientific and helping the business achieve effective operation

Analysis refers to breaking a whole into its separate components for individual examination. Data analysis is a [process](https://en.wikipedia.org/wiki/Process_theory) for obtaining raw data and converting it into information useful for decision-making by users. Data are collected and analyzed to answer questions, test hypotheses or disprove theories

**2.2 Data analysis**

There are total five steps to analyze a data set

1. Hypothesis generation
2. Data exploring
3. Data cleaning
4. Feature engineering
5. Model building

**Hypothesis generation:**

A statistical hypothesis, sometimes called confirmatory data analysis, is a [hypothesis](https://en.wikipedia.org/wiki/Hypothesis) that is testable on the basis of [observing](https://en.wikipedia.org/wiki/Observable_variable) a process that is [modeled](https://en.wikipedia.org/wiki/Statistical_model) via a set of [random variables](https://en.wikipedia.org/wiki/Random_variable).A statistical hypothesis test is a method of [statistical inference](https://en.wikipedia.org/wiki/Statistical_inference). Commonly, two statistical data sets are compared, or a data set obtained by sampling is compared against a synthetic data set from an idealized model. A hypothesis is proposed for the statistical relationship between the two data sets, and this is compared as an [alternative](https://en.wikipedia.org/wiki/Alternative_hypothesis) to an idealized null hypothesis that proposes no relationship between two data sets. The comparison is deemed [statistically significant](https://en.wikipedia.org/wiki/Statistically_significant) if the relationship between the data sets would be an unlikely realization of the [null hypothesis](https://en.wikipedia.org/wiki/Null_hypothesis) according to a threshold probability—the significance level. Hypothesis tests are used when determining what outcomes of a study would lead to a rejection of the null hypothesis for a pre-specified level of significance.

Understanding the problem better by brainstorming possible factors that can impact the outcome.

**Data exploring:**

Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a data set, including its size, accuracy, initial patterns in the data and other attributes. It is commonly conducted by data analysts using visual analytics tools, but it can also be done in more advanced statistical software, like python and R.

Before it can conduct [analysis](https://whatis.techtarget.com/definition/statistical-analysis) on data collected by multiple data sources and stored in data warehouses, an organization must know how many cases are in a data set, what variables are included, how many missing values there are and what general hypotheses the data is likely to support. An initial exploration of the data set can help answer these questions by familiarizing analysts with the data with which they are working.

Looking at categorical and continuous feature summaries and making inferences about the data.

**Data cleaning:**

**Data cleansing** or **data cleaning** is the process of detecting and correcting (or removing) corrupt or inaccurate [records](https://en.wikipedia.org/wiki/Storage_record) from a record set, [table](https://en.wikipedia.org/wiki/Table_(database)), or [database](https://en.wikipedia.org/wiki/Database) and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the [dirty](https://en.wikipedia.org/wiki/Dirty_data) or coarse data. Data cleansing may be performed [interactively](https://en.wikipedia.org/wiki/Interactively) with [data wrangling](https://en.wikipedia.org/wiki/Data_wrangling) tools, or as [batch scripting processing](https://en.wikipedia.org/wiki/Batch_processing).

So, cleaning is the part of imputing missing values in the data and checking for outliers.

**Feature Engineering:**

Feature engineering is the process of using [domain knowledge](https://en.wikipedia.org/wiki/Domain_knowledge) of the data to create [features](https://en.wikipedia.org/wiki/Feature_(machine_learning)) that make [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive.

Feature engineering is an essential part of building any intelligent system. Even though you have a lot of newer methodologies coming in like deep learning and meta-heuristics which aid in automated machine learning, each problem is domain specific and better features (suited to the problem) is often the deciding factor of the performance of your system. Feature Engineering is an art as well as a science and this is the reason Data Scientists

Often spend 70% of their time in the data preparation phase before modeling.

So, Feature Engineering is modifying existing variables and creating new ones for analysis

**Model Building:**

Machine learning happens to be a small part of this process. The model building process involves setting up ways of collecting data, understanding and paying attention to what is important in the data to answer the questions you are asking, finding a statistical, mathematical or a simulation model to gain understanding and make predictions.

All of these things are equally important and model building is a crucial skill to acquire in every field of science. The process stays true to the scientific method, making what you learn through your models useful for gaining an understanding of whatever you are investigating as well as make predictions that hold true to test.

**CHAPTER THREE**

**TOOLS**

**3.1 Software needed to run this project**

1. Python Version 3.5 (As programming language)
2. Numpy
3. Pandas
4. Anaconda promt (Package management)
5. Spyder (IDE- Integrated Drive Electronics)

**3.2 Data analysis languages**

**Python:**

It is said that there is no short cut to success but if you are a quick learner and want to get up and running with a widely used, easy to learn programming language, [PYTHON](https://www.livewireindia.com/python_software_training.php) is what you are looking at. Its USP is readability and compactness. It enables programmers to express same concepts in shorter code fragments. Developers coming from diverse programming backgrounds and used to different styles (object oriented, imperative, functional, procedural) find it easy to adapt to Python. It allows easier scalability which makes it equally suited to handle small scale and large scale applications.



Applications such as Pinterest and Instagram are built using Python. It is rapidly gaining popularity at the academic level and finds itself amongst the most commonly taught programming language in the schools.

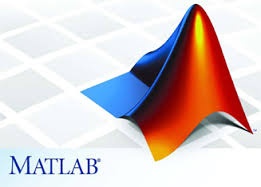
**R:**

R is a considerable deviation from the languages we have discussed so far. It is not a substitute for any of the languages we’ve already discussed. R is essentially a dedicated language for statistical computing and graphics. Given the way data is being generated in the 21st century, R has become the favorite language for data analysts and scientists around the world. R has been ranked at No. 6 in the IEEE’s Top 10 Programming Language of 2015 and with the growing influence of Big Data and emergence of Internet of Things, you can be assured that it will continue to be a hot skill for years to come, and beyond.



**Mat Lab:**

[Mat lab](https://www.livewireindia.com/electronic-design-automation.php) is a must learn programming language for data science, particularly for working with matrixes. [Mat lab](https://www.livewireindia.com/electronic-design-automation.php) is not an open source language but is used extensively in academic courses because of its suitability for mathematical modelling and data acquisition. Though [Mat lab](https://www.livewireindia.com/electronic-design-automation.php) lacks the volume of open source community driven support, its extensive adoption in academic courses has made it popular for data science. MATLAB programming language is good for data science tasks that involve linear algebraic computations, simulations and matrix computations. LAPACK and BLAS libraries for matrix multiplication in MATLAB are highly optimized that speed up execution. However, MATLAB imposes restrictions on code portability (ability to run code on other computer). Data scientists can run compiled application on other computer using the MCR (MATLAB Component Runtime) components, but the app must have the same version of MCR installed. Crowd Flower analysis shows that 10%-15% of data scientist job listing require this [Mat lab](https://www.livewireindia.com/electronic-design-automation.php) programming skill.



**Scala:**

According to the O’Reily 2015 Data Science Salary Survey, the use of Scala programming language for data science increased by 10% in 2016. Scala programming language is a fusion of object oriented and functional programming languages that helps build robust and scalable data science applications. As organizations aspire to work with growing amount of real-time data, Scala programming language helps data scientists write short and expressive code whilst delivering high performance and type safe applications that are impressive and valuable. Scala for data science requires a little extra knack of abstraction and thinking. However, once a data scientist becomes familiar with its high level functional features, productivity boosts dramatically. Scalability and number crunching abilities of Scala have made it one among the best programming languages for data science.

Most of the data science projects today follow an agile methodology, data scientists want to change the requirements of the code as they perform data explorations so that they can adjust them at each iteration. Usually, data scientists first write some code with associated tests and then after the tests are complete, the APIs are broken. Every time a data scientist performs refactoring, there is a probability of introducing new bugs and wordlessly breaking the previous coding logic. Scala being a compile language has better advantage in terms of safe refactoring over other data science programming languages like Python.



**3.3 Why Python:**

Python is an interpreted, dynamically-typed language with a precise and efficient syntax. Python has a good REPL and new modules can be explored from the REPL with dir() and docstrings. That's one reason to prefer Python over C, C++, or Java.  
  
The Python community invested in the mid-1990s in **Numeric**, an "extension to Python to support numeric analysis as naturally as Mat lab does”. Numeric later evolved into **NumPy**. Several years later, the plotting functionality from Mat lab was ported to Python with **matplotlib**. Libraries for scientific computing were built around NumPy and matplotlib and bundled into the **SciPy** package, which was commercially supported by Enthought. Python's support for Mat lab-like array manipulation and plotting is a major reason to prefer it over Perl and Ruby.  
  
Today, the most popular alternatives to Python for data scientists are R, Mat lab/Octave, and Mathematica/Sage. In addition to the work mentioned above to port features from Mat lab into Python, recent work has ported several popular features from R and Mathematica into Python.  
  
From R, the data frame and associated manipulations (from the plyr and reshape packages) have been implemented by the **pandas** library. The **scikit-learn** project presents a common interface to many machine learning algorithms, similar to the caret package in R.  
  
From Mathematica/Sage, the concept of a "notebook" has been implemented with **IPython notebooks**.

**3.4 Numpy**

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

* a powerful N-dimensional array object
* sophisticated (broadcasting) functions
* tools for integrating C/C++ and Fortran code
* useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the [BSD license](http://www.numpy.org/license.html#license), enabling reuse with few restrictions.

**3.5 Pandas**

Python has long been great for data munging and preparation, but less so for data analysis and modeling. pandas helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R.

Combined with the excellent [IPython](https://ipython.org/) toolkit and other libraries, the environment for doing data analysis in Python excels in performance, productivity, and the ability to collaborate.

Pandas does not implement significant modeling functionality outside of linear and panel regression; for this, look to [stats models](http://statsmodels.sf.net/) and [scikit-learn](http://scikit-learn.org/). More work is still needed to make Python a first class statistical modeling environment, but we are well on our way toward that goal.

* A fast and efficient **Data Frame** object for data manipulation with integrated indexing;
* Tools for **reading and writing data** between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
* Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
* Flexible **reshaping** and pivoting of data sets;
* Intelligent label-based **slicing**, **fancy indexing**, and **sub setting** of large data sets;
* Columns can be inserted and deleted from data structures for **size mutability**;
* Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;
* High performance **merging and joining** of data sets;
* **Hierarchical axis indexing** provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
* **Time series**-functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
* Highly **optimized for performance**, with critical code paths written in [Cython](http://www.cython.org/)or C.
* Python with pandas is in use in a wide variety of **academic and commercial** domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

**3.6 Anaconda**

The open-source [Anaconda Distribution](https://docs.anaconda.com/anaconda/) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine,

Enabling individual data scientists to:

* Quickly download 1,500+ Python/R data science packages
* Manage libraries, dependencies, and environments with [Conda](https://conda.io/docs/)
* Develop and train machine learning and deep learning models with [scikit-learn](https://scikit-learn.org/stable/), [TensorFlow](https://www.tensorflow.org/), and [Theano](https://pypi.org/project/Theano/)
* Analyze data with scalability and performance with [Dask](https://dask.org/), [NumPy](http://www.numpy.org/), [pandas](https://pandas.pydata.org/), and [Numba](http://numba.pydata.org/)
* Visualize results with [Matplotlib](https://matplotlib.org/), [Bokeh](https://bokeh.pydata.org/en/latest/), [Datashader](http://datashader.org/), and [Holoviews](http://holoviews.org/)

**3.7 Spyder**

Spyder is an [open source](https://en.wikipedia.org/wiki/Open-source_software) cross-platform [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for scientific programming in the [Python language](https://en.wikipedia.org/wiki/Python_(programming_language)). Spyder integrates with a number of prominent packages in the scientific Python stack, including [NumPy](https://en.wikipedia.org/wiki/NumPy), [SciPy](https://en.wikipedia.org/wiki/SciPy), [Matplotlib](https://en.wikipedia.org/wiki/Matplotlib), [pandas](https://en.wikipedia.org/wiki/Pandas_(software)), [IPython](https://en.wikipedia.org/wiki/IPython), [SymPy](https://en.wikipedia.org/wiki/SymPy) and [Cython](https://en.wikipedia.org/wiki/Cython), as well as other open source software. It is released under the [MIT license](https://en.wikipedia.org/wiki/MIT_license).

Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first- and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, [Pylint](https://en.wikipedia.org/wiki/Pylint) and Rope. It is available cross-platform through [Anaconda](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)), on Windows, on macOS through [MacPorts](https://en.wikipedia.org/wiki/MacPorts" \o "MacPorts), and on major Linux distributions such as [Arch Linux](https://en.wikipedia.org/wiki/Arch_Linux), [Debian](https://en.wikipedia.org/wiki/Debian" \o "Debian), [Fedora](https://en.wikipedia.org/wiki/Fedora_(operating_system)), [Gentoo Linux](https://en.wikipedia.org/wiki/Gentoo_Linux), [openSUSE](https://en.wikipedia.org/wiki/OpenSUSE" \o "OpenSUSE) and [Ubuntu](https://en.wikipedia.org/wiki/Ubuntu_(operating_system)).

Spyder uses [Qt](https://en.wikipedia.org/wiki/Qt_(software)" \o "Qt (software)) for its GUI, and is designed to use either of the [PyQt](https://en.wikipedia.org/wiki/PyQt" \o "PyQt) or [PySide](https://en.wikipedia.org/wiki/PySide" \o "PySide) Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backen.

**CHAPTER FOUR**

**CODING AND ANALYSIS**

**4.1 Stages**

We will explore the problem in following stages:

1. Hypothesis Generation
2. Data Exploration
3. Data Cleaning
4. Feature Engineering
5. Model Building

**4.2 Hypothesis Generation**

Understanding the problem statement is the first and foremost step

**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.

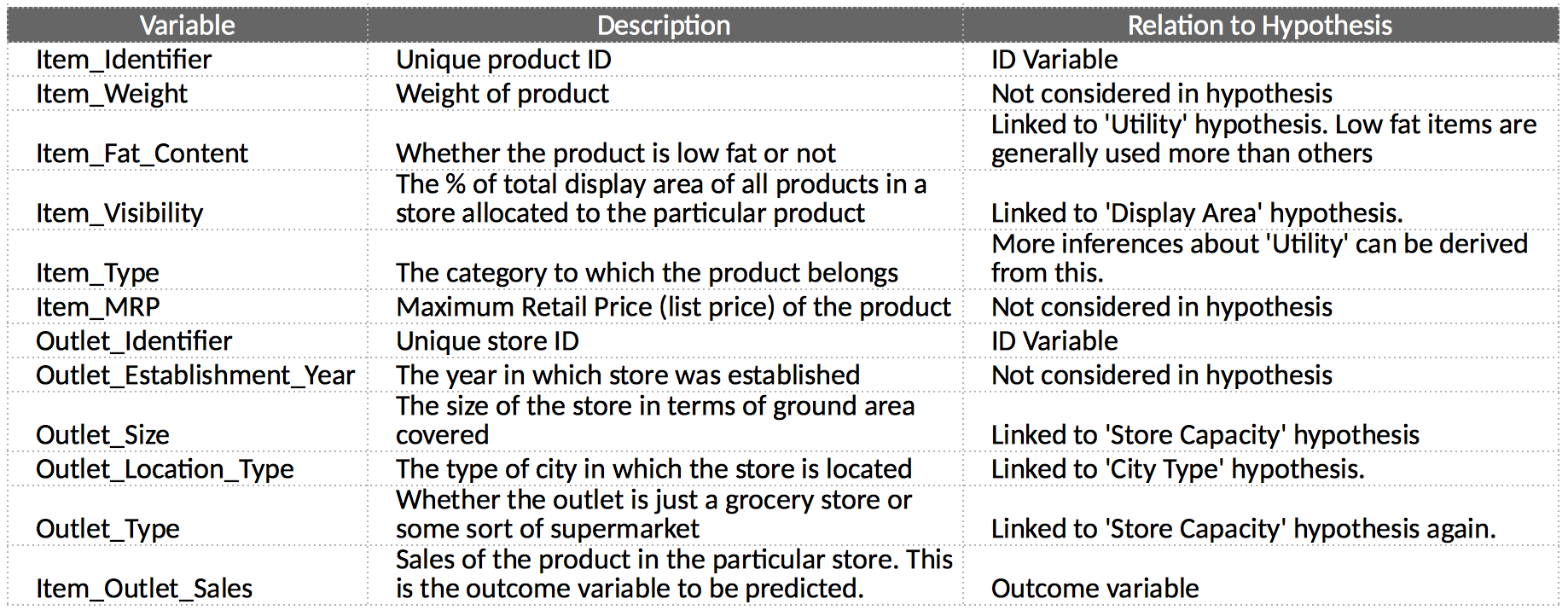
**Product Level Hypotheses:**

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.

**4.3 Data Exploration**

We’ll be performing some basic data exploration here and come up with some inferences about the data. We’ll try to figure out some irregularities and address them in the next section.

The first step is to look at the data and try to identify the information which we hypothesized vs the available data. A comparison between the data dictionary on the competition page and out hypotheses is shown below:

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/02/0.-data-dictionary-1.png)

Here to mention that our datasets are

test.csv

train.csv

It’s generally a good idea to combine both train and test data sets into one, perform feature engineering and then divide them later again. This saves the trouble of performing the same steps twice on test and train. Let’s combine them into a data frame ‘data’ with a ‘source’ column specifying where each observation belongs.

Some observations:

1. **Item\_Visibility** has a min value of zero. This makes no practical sense because when a product is being sold in a store, the visibility cannot be 0.
2. **Outlet\_Establishment\_Years** vary from 1985 to 2009. The values might not be apt in this form. Rather, if we can convert them to how old the particular store is, it should have a better impact on sales.
3. The lower ‘count’ of Item\_Weight and Item\_Outlet\_Sales confirms the findings from the missing value check.

**4.4 Data Cleaning**

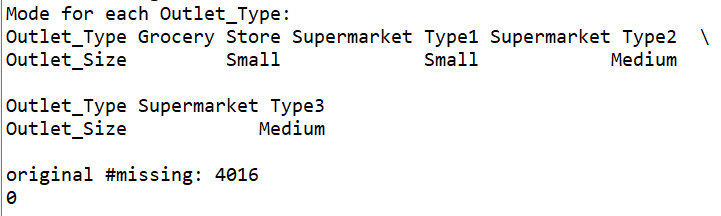
This step typically involves imputing missing values and treating outliers. Though outlier removal is very important in regression techniques, advanced tree based algorithms are impervious to outliers. So I’ll leave it to you to try it out. We’ll focus on the imputation step here, which is a very important step.

**Imputing Missing Values**

We found two variables with missing values – Item\_Weight and Outlet\_Size. Let’s impute the former by the average weight of the particular item

****

This confirms that the column has no missing values now. Let’s impute Outlet\_Size with the mode of the Outlet\_Size for the particular type of outlet.



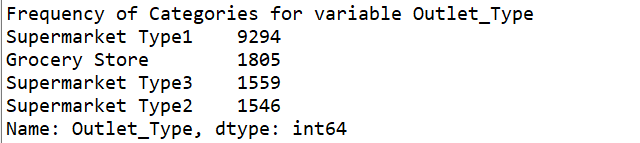
This confirms that there are no missing values in the data. Let’s move on to feature engineering now.

**4.5 Feature Engineering**

We explored some nuances in the data in the data exploration section. Let’s move on to resolving them and making our data ready for analysis. We will also create some new variables using the existing ones in this section.

### Step 1: Consider combining Outlet\_Type

During exploration, we decided to consider combining the Supermarket Type2 and Type3 variables. But is that a good idea? A quick way to check that could be to analyze the mean sales by type of store. If they have similar sales, then keeping them separate won’t help much.



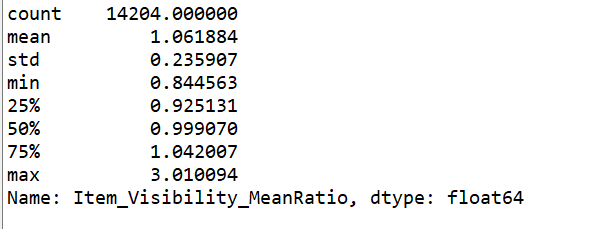
### Step 2: Modify Item\_Visibility

### We noticed that the minimum value here is 0, which makes no practical sense. Let’s consider it like missing information and impute it with mean visibility of that product.



So we can see that there are no values which are zero.

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.



Thus the new variable has been successfully created.

### Step 3: Create a broad category of Type of Item

### Earlier we saw that the Item\_Type variable has 16 categories which might prove to be very useful in analysis. So it’s a good idea to combine them. One way could be to manually assign a new category to each. But there’s a catch here. If you look at the Item\_Identifier, i.e. the unique ID of each item, it starts with either FD, DR or NC. If you see the categories, these look like being Food, Drinks and Non-Consumables. So I’ve used the Item\_Identifier variable to create a new column:

### 

### Step 4: Determine the years of operation of a store

### We wanted to make a new column depicting the years of operation of a store

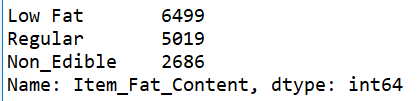
### 

### Step 5: Modify categories of Item\_Fat\_Content

### We found typos and difference in representation in categories of Item\_Fat\_Content variable

### 

In step 4 we saw there were some non-consumables as well and a fat-content should not be specified for them. So we can also create a separate category for such kind of observations.



### Step 6: Numerical and One-Hot Coding of Categorical variables

### Since scikit-learn accepts only numerical variables, I converted all categories of nominal variables into numeric types. Also, I wanted Outlet\_Identifier as a variable as well. So I created a new variable ‘Outlet’ same as Outlet\_Identifier and coded that.

### One-Hot-Coding refers to creating dummy variables, one for each category of a categorical variable. 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 ‘get\_dummies’ function of Pandas

### 

### Step 7: Exporting Data

### Final step is to convert data back into train and test data sets. It’s generally a good idea to export both of these as modified data sets so that they can be re-used for multiple sessions.

## **4.6** **Model Building**

**4.6.1 Mean based Model**

Taking overall mean is just the simplest way

#Mean based:

mean\_sales = train['Item\_Outlet\_Sales'].mean()

#define a dataframe with IDs for submission:

base1 = test[['Item\_Identifier','Outlet\_Identifier']]

base1.is\_copy = None

base1['Item\_Outlet\_Sales'] = mean\_sales

#Export submission file

base1.to\_csv("alg0.csv", index = False)

### 4.6.2 Linear Regression Model

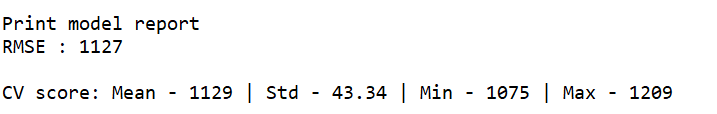
### Let’s make our first linear-regression model.

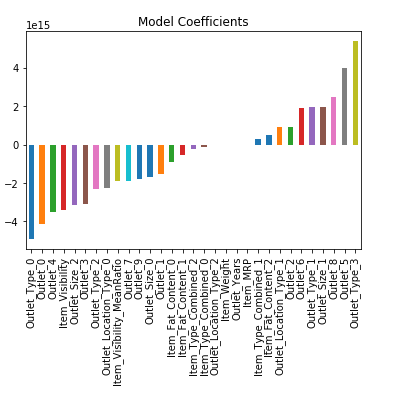
Linear Regression is used for predictive analysis. It is a technique which explains the degree of relationship between two or more variables (multiple regression, in that case) using a best fit line / plane. Simple Linear Regression is used when we have, one independent variable and one dependent variable.

Regression technique tries to fit a single line through a scatter plot (see below).  The simplest form of regression with one dependent and one independent variable is defined by the formula:

Y = aX + b

Source: Wikipedia





### Ridge Regression Model:

### We can see this is better than baseline model. But if you notice the coefficients, they are very large in magnitude which signifies overfitting

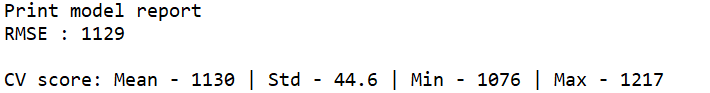
Ridge and Lasso [regression](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2?utm_source=blog&utm_medium=RideandLassoRegressionarticle)are powerful techniques generally used for creating parsimonious models in presence of a ‘large’ number of features. Here ‘large’ can typically mean either of two things:

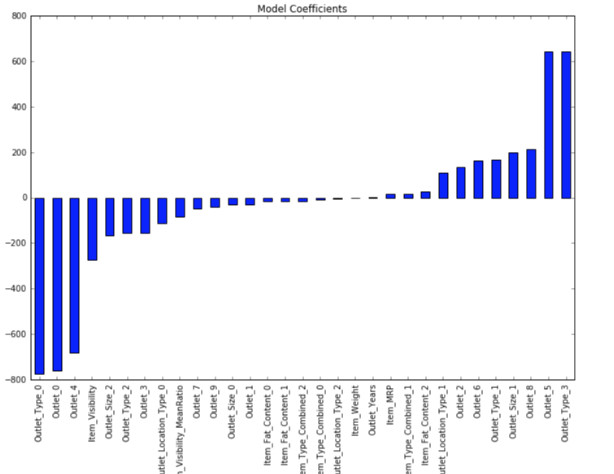
1. Large enough to enhance the tendency of a model to overfit (as low as 10 variables might cause overfitting)
2. Large enough to cause computational challenges. With modern systems, this situation might arise in case of millions or billions of features

Though Ridge and Lasso might appear to work towards a common goal, the inherent properties and practical use cases differ substantially. If you’ve heard of them before, you must know that they work by penalizing the magnitude of coefficients of features along with minimizing the error between predicted and actual observations. These are called ‘regularization’ techniques. The key difference is in how they assign penalty to the coefficients:

1. **Ridge Regression:**
   * Performs L2 regularization, i.e. adds penalty equivalent to **square of the magnitude** of coefficients
   * Minimization objective = LS Obj + α \* (sum of square of coefficients)

Source: internet





**CHAPTER FIVE**

**DISCUSSION**

**5.1 Output Discussion:**

### From our last model we have used Ridge Regression Model

And to recall that Ridge and Lasso [regression](https://courses.analyticsvidhya.com/courses/introduction-to-data-science-2?utm_source=blog&utm_medium=RideandLassoRegressionarticle)are powerful techniques generally used for creating parsimonious models in presence of a ‘large’ number of features. Here ‘large’ can typically mean either of two things:

1. Large enough to enhance the tendency of a model to overfit (as low as 10 variables might cause overfitting)
2. Large enough to cause computational challenges. With modern systems, this situation might arise in case of millions or billions of features

So, basically the Ridge Regression is Linear regression with large number of feature

Now let’s see the output again

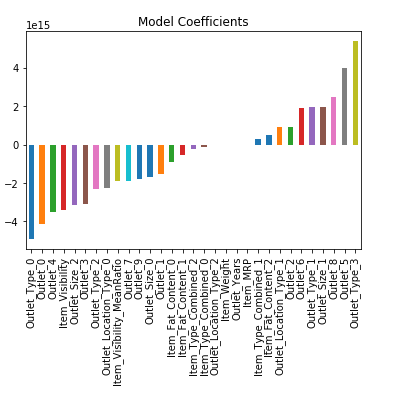


Fig: Linear Regration

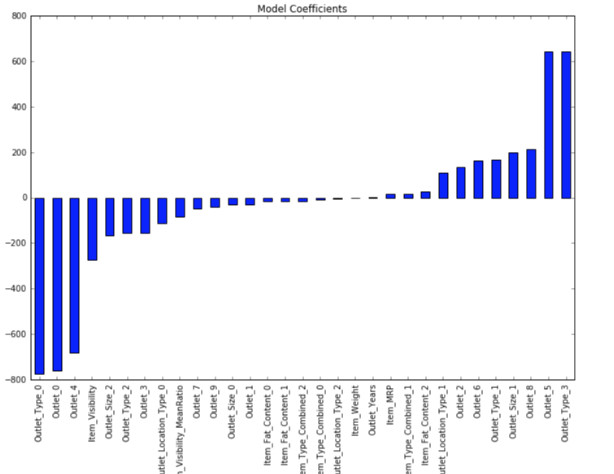
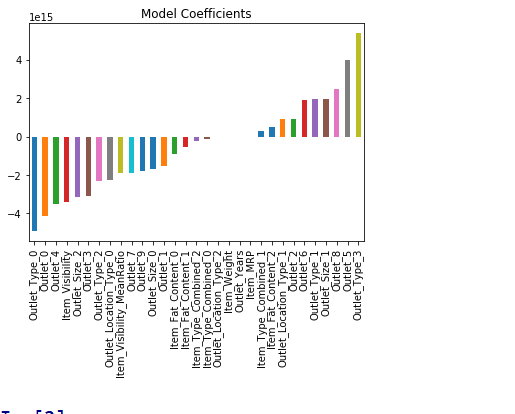


Fig: Ridge Regression Model



Here the outlet\_7, Outlet\_9, Outlet\_size\_0, Item\_fat\_content\_0, Item\_fat\_content\_1, Outlet\_location\_type\_2 has less deviation in test and train data set.

So, the shop 7, shop 9, Item that contains low and regular fat, shop\_location\_type\_2, supermarket with small size has fitted well in the test data line.

So, these reasons does not affect much in selling products

But Outlet\_type\_3 (large supermarket), Out\_let\_5, Outlet\_8 (shop 5, 8), outlet\_size\_1(medium),Outlet\_type\_1 has positive effect on selling and prediction is that these can increase the sales.

On the other hand outlet\_type\_0, Outlet\_0, Outlet\_4 (Shop 0, 4) Item\_size\_2 (large) Outlet\_size\_2 (Medium) Outlet\_location\_type\_0 has negative effect on selling and my prediction is that these can decrease the sales.

CHAPTER SIX

CONCLUSION

**6.1 Conclusion**

To improve this analysis farther we can use some more model

1. Decision tree model
2. Random forest model

These model may decrease the deviation and help to make better decision.