STUDY OF A FEW DATA MINING ALGORITHMS

Md. Shahadat Hossain	**14.01.04.062**
Abrar Akhyer Abir	**14.01.04.069**
Muhammad Muntasir Mamoor	**14.01.04.072**
Md. Hasan Shahriar	**14.01.04.096**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY
DHAKA, BANGLADESH

MAY 2018

STUDY OF A FEW DATA MINING ALGORITHMS

A Project and Thesis Report
Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering

By

Md. Shahadat Hossain	14.01.04.062
Abrar Akhyer Abir	14.01.04.069
Muhammad Muntasir Mamoor	14.01.04.072
Md. Hasan Shahriar	14.01.04.096

Supervised by
Dr. S. M. Abdullah Al-Mamun
Professor
Department of Computer Science and Engineering

AHSANULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY
DHAKA, BANGLADESH
MAY 2018

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of Dr. S. M. Abdullah Al-Mamun, Professor, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE4100: Project & Thesis I and CSE4250: Project & Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

……...	……..
Md. Shahadat Hossain	Abrar Akhyer Abir
……	……..
Muhammad Muntasir Mamoor	Md. Hasan Shahriar

Counter Signature:

……..……..

Dr. S. M. Abdullah Al-Mamun
Professor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGMENT

First of all, we would like to express our gratitude to Almighty Allah for successful completion of our project and thesis work. We have many people to thank for the help and encouragement regarding this work. We would like to thank our supervisor, Professor Dr. S. M. Abdullah Al-Mamun, Department of CSE, AUST, for his precious advice and support during this study. With his guidance, we have been able to finish this work on time. We would also like to thank all the faculty members of the department who have helped us in a various way during the period.

Thankfully,

Md. Shahadat Hossain
Abrar Akhyer Abir
Muhammad Muntasir Mamoor
Md. Hasan Shahriar

ABSTRACT

The aim of our work is to study and understand different types of data mining algorithms. These algorithms have been applied on two transactional datasets to compare and analyze their performances. We have examined different popular data mining techniques and designed a basic data warehouse to simplify the datasets. Different application issues have been discussed after applying the algorithms on simplified warehoused data. We have improved the performance of the algorithms by tuning some parameters, using variable transformation and outlier analysis. Comparative performance of the algorithms has also been visualized.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

1.1 Introduction

From the very beginning of human evolution, people started to observe and learn the environment. Human brain saved all the observations into its memory system. With the passage of time, they adapted themselves to the environment based on their previous observations. Each distinct observation can be analogized as data. From then to now the increasing exponentially of data generation has been continued. After the industrial revolution, all the industrial operations started to generate data at high scales. These huge amounts of unstructured data became valuable to the authorities. From then data management becomes a part of the study. With the help of increased computational ability, different complex data management and maintenance techniques were invented and became popular to the computer science community. During 1990, internet became popular and internet-based applications & users started to generate huge data than before. The importance of modern data management and analysis have become more essential than before. Modern business processes make billions of transactions per seconds and create an enormous amount of data which are important to understanding customer mode, business growth, sustainability, etc. To deal with such data and perform efficient analysis, a new branch of computation is created which is called data mining. Artificial Intelligence has added a new dimension in the data mining sector. Difficulties of Conventional processes are being solved by Artificial Intelligence and other related technologies.

1.2 Introduction to our work

We have aimed to study on different mining algorithms and apply them to understand their performance. For this, we have selected two opensource datasets. The first dataset is taken from Tableau Community [12]. It is about global superstore transactions over the period of 2011 to 2015. The dataset contains 26 attributes and more than fifty-one thousands of records. We have planned to predict different measures based on the interrelated data and perform the actions via different algorithms and record their performance. For this we have selected **Python** as the programming language; **Scikit-Learn,** a Python-based machine learning library; **Pandas**, a data manipulation library; different editors and IDEs.

1.3 Problem Definition

The first and foremost step of any work is the complete and clear problem definition. In our case, we need to state what we really want to do. But that is not enough. More precisely, we must ask the right question with proper details. Then we can find a solution.

In our work, initially, we have to analyze the datasets and create a structured format for them to apply algorithms. To understand historical data organization and classic data mining processes, we have planned to design a data warehouse to store organized data. Then we will apply algorithms on the warehouse data. But there are different varieties of each algorithm and performance of particular algorithms varies in accordance with its application. So, we will test the performance of the algorithms by finding the accuracy rate on different applications and then show a comparative performance chart.

1.4 The process of the solution

We have segmented our solutions into different processes which are given below:

i. Data Collection
ii. Data pre-processing
iii. Algorithm Selection
iv. Important Feature selection for each algorithm
v. Model Creation
vi. Performance Testing
vii. Comparison and visualization of algorithms.

i. Data Collection

First and foremost step of our solution is to collect the data to perform all the operations. For this, we have collected two datasets. An overview of data collection is given in the introduction section of this report.

ii. Data Pre-processing

When the data collection is completed, the second step is to process the dataset to perform further operations. We have designed a basic data warehouse to process the dataset where the dataset is normalized and segmented into different dimension tables and a fact table. Detailed descriptions of this step are discussed in the data warehousing section of this report.

iii. Algorithm Selection

The most important process of our work is to select the appropriate algorithms to perform operations on the datasets. We have selected different algorithms for different operations.

iv. Important Feature selection

For a specific algorithm, we have to select relevant features to perform the operations. There are many approaches for selecting best features. Some of the techniques are discussed later in this report.

v. Model Creation

When the features/ parameters are ready, the next step is to make the mathematical model for the algorithms. In this step, the mathematical expression of the algorithms is developed.

vi. Performance Testing

After the mathematical model is created, we have tested the model whether it works or not and provides the desired solution.

vii. Comparison and visualization of algorithms

Finally, we have to compare the results of the algorithms for a specific operation and determine which algorithm works better for that specific operation and visualize the comparison diagram.

1.5 Goal of our work

The goal of our work is simple. We will study different data mining techniques and few algorithms to understand their scopes, variations, relevancy and limitations. For this purpose, we have designed a data warehouse from one of our datasets and performed other operations which will be discussed later in this report.

1.6 Outline of the report

This section gives an overview of the contents of our report. At the very beginning of this book, Abstract of our work has been given.

In chapter one, we have provided a brief introduction to our work, goals and problem definition.

In chapter two, we put the literature review of our work where basic concepts and historical background of data mining and warehousing were discussed.

In chapter three, an overview of our targeted algorithms are enlisted. We have also put the necessary mathematical expressions of the algorithms.

Chapter four is about the design and implementation of data warehouse. A snowflake schema and data dictionary of our data warehouse are also described there.

In chapter five, we started with the implementation methodologies of our algorithms and then we described our total implementation techniques.

Chapter six is about the comparative result analysis of our clustering and regression algorithms and dataset visualization.

We have discussed the scope of further work in chapter seven. Finally, we have concluded our report with a formal conclusion. Reference and appendix were given at the end of this book.

Chapter 2
Literature Review

2.1 Literature Review

To understand and gain knowledge on any topic, literature is the best reliable source to start with. It gives an in-depth overview of any topic. There are many researchers who have worked on data mining before and their works are recognized globally. To start and continue our work we have studied few books and references of such researchers.

2.2 Historical background of Data Warehouse & Mining

Jaiwei Han et al. [1] pointed that since 1960, systematic database and information technology has been evolved from the primitive file processing system to sophisticated database system. In 1970 the database system progressed from hierarchical and network database system to relational database. From then users gained convenient flexible data access through query languages and user interface. After the establishment of database management systems, database technology moved toward the development of advanced database systems, data warehousing & data mining for advanced data analysis and web-based databases during mid-1980. Advance data analysis started from 1980. Data warehousing techniques were introduced and from 1990, when the internet started, huge volumes of data have been accumulated beyond databases and data warehouses and the trend of data mining started to popular to the community.

2.3 Data Mining

S.B Soumya et al. [2] defined data mining as the extraction of patterns representing knowledge implicitly stored or captured in large databases, data warehouses, the Web, other massive information repositories or data streams.

From other two references, we can define data mining as the computing process of discovering patterns in large data set involving methods at the intersection of machine learning, statistics and database system [3]. It is an essential process where intelligent methods are applied to extract data patterns [1][3].

It exists, however, in many variations on this theme, such as the Cross Industry Standard Process for Data Mining (CRISP-DM) which defines six phases [2]:

i. Business Understanding
ii. Data Understanding
iii. Data Preparation
iv. Modeling
v. Evaluation
vi. Deployment

i. Business Understanding

Understand the project objectives and requirements from a business perspective and then convert this knowledge into a data mining problem definition and a preliminary plan designed to achieve the objectives.

ii. Data Understanding

This is one of the important steps in data mining. Start by collecting data, then get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses about hidden information. Understanding these things will give us a clear view of the nature of data.

iii. Data Preparation

Includes all activities required to construct the final dataset. We must design our dataset in such a way that will be fed into the modeling tool from the initial raw data. Tasks include tables, case and attribute selection as well as transformation and data cleaning, deal with missing values, noisy data and correlated columns.

iv. Modeling

Modeling is an important part of our research. Select and apply a variety of modeling techniques and calibrate tool parameters to optimal values are the main activities of this phase. Typically, there are several techniques for the same data mining problem type. Some techniques have specific requirements in the form of data. Therefore, stepping back to the data preparation phase is often needed.

v. Evaluation

Thoroughly evaluate the model and review the steps executed to construct the model, to be certain it properly achieves the business objectives. Determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the data mining results is reached.

vi. Deployment

Organize and present the results of data mining. Deployment can be as simple as generating a report or as complex as implementing a repeatable data mining process.

2.4 Data Warehouse for Data Mining

A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transaction data, but it can include data from other sources. It separates analysis workload from transaction workload and enables an organization to consolidate data from several sources [1]. According to William H. Inmon, a leading architect in the construction of Data Warehouse systems, a data warehouse is a subject-oriented, integrated, time-variant and nonvolatile collection of data in support of management's decision-making process. In this definition, we get four keywords: **subject-oriented, integrated, time-variant and Nonvolatile** which distinguish the Data warehouse from other Data repository systems. Subject-oriented means Data warehouses are designed to help you analyze data. For example, to learn more about your company's sales data, you can build a warehouse that concentrates on sales. Using this warehouse, you can answer questions like "Who was our best customer for this item last year?" This ability to define a data warehouse by subject matter, sales, in this case, makes the data warehouse subject oriented. By the word Integrated means, DW integrates current and historical data from different sources. Time-variant specifies to discover trends in business, analysts need large amounts of data. This is very much in contrast to online Transaction Process System (OLTP) systems, where performance requirements demand that historical data be moved to an archive. A data warehouse's focus on change over time is what is meant by the term time variant. Non-volatile collection of data Nonvolatile means that once entered the warehouse, data should not change. This is logical because the purpose of a warehouse is to enable you to analyze what has occurred.

Data Warehouse Application on Data mining [1]:

Data warehousing is particularly important for data mining for the following reasons:

i. The high quality of data in data warehouses:

Most data mining tools need to work on integrated, consistent and cleaned data, which requires costly data cleaning, data integration and data transformation as preprocessing steps. A data warehouse constructed by such preprocessing serves as a valuable source of high-quality data for OLAP as well as for data mining.

ii. Exploration of multidimensional data:

Effective data mining needs exploratory data analysis. A user will often want to traverse through a database, select portions of relevant data, analyze them at different granularities and present knowledge results in different forms.

2.5 Techniques involves in data mining

Several techniques have been implemented to find insights from data and day by day these practices became more mature. Scientists invent new techniques for optimizing computational cost and maximizing performance based on data and finding patterns. In our work, we have chosen to work with numeric transactional data that contains some sensitive information. Nowadays machine learning becomes a buzzword in the Artificial Intelligence and Data analysis industry. This refers to the autonomous working capability of a program based on given data and learning technique. Here, learning techniques indicate mathematical models to perform operations on given dataset.

We have implemented four famous and easy algorithms to continue our task. Few background studies of these algorithms are given here:

2.5.1 Linear Regression

It is one of the supervised learning algorithms. In statistics, Linear regression is an approach to modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variable) denotes X. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable the process is called multiple linear regression [14]. This is an easy algorithm to

implement. In Linear regression, the relationship is modeled using linear predictor functions whose unknown model parameters are estimated from data. Such models are called linear models [15].

There may have different linear models. Most commonly used linear models are:

- Simple and Multiple regression
- Generalized Linear model
- Heteroscedastic models
- Hierarchical Linear models

Performance of the algorithm varies from estimation methods. There are different estimation strategy and each strategy may have different methods. Few of them are given below:

- Least Square estimation and related topics
    - Ordinary Least Square
    - Generalized least Square
    - Percentage least square
    - Iteratively reweighted least square
    - Instrumental Variable
    - Optimal Instrument regression
    - Total Least Square
- Maximum Likelihood estimation and related techniques
    - Maximum Likelihood estimation
    - Ridge regression
    - Least Absolute Deviation
    - Adaptive estimation
- Bayesian Linear Regression
- Quantile Regression
- Mixed Models
- Principal Component Regression
- Least Angle Regression
- Theil Sen estimator
- α-trimmed mean

2.5.2 K Nearest Neighbor

K Nearest Neighbor algorithm is known KNN in short. It is a supervised classification algorithm. It can be used as both classification and regression. It is very easy to implement but computationally very costly. We will discuss the pros and cons of KNN in the algorithm section. In KNN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor. A non-parametric equation is used to model KNN in both cases [16].

In classification, K is used defined constant. And an unlabeled vector (test data points) is classified by assigning the label which is most frequent among the k training samples nearest to the test sample. Different Distance matrices are used to measure the distance. Most common distance equation is Euclidian distance. For text classification Overlap metric or Hamming distance is considered as best fit distance equation. Manhattan distance can also be used in classification.

2.5.3 Polynomial Regression

In statistics, polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modeled as an n^{th} degree polynomial in x. It fits a non-linear relationship between the value of x and the corresponding conditional mean of y, denoted E(y|x) and used to describe nonlinear phenomena.

Though polynomial regression fits a non-linear model, as a statistical estimation problem it is linear. In the sense that the regression function E(y|x) is linear in the unknown parameters that are estimated from the data. For this reason, polynomial regression is a special case of multiple linear regression.

Chapter 3
Analysis of the Selected Algorithms

3.1 Reason for Selecting Particular Algorithms

We have selected four Data Mining algorithms to study. They are:

1. Linear Regression,
2. Polynomial Regression,
3. Logistic Regression &
4. K-Nearest Neighbor.

Here linear & polynomial regressions are regression algorithms and logistic regression, K-Nearest Neighbor are classification algorithm. We are working mostly on transactional data. Most of the attributes are numeric. For numeric data, these four algorithms give better performance. Beside these are the fundamental Data mining algorithms. Details of these algorithms and implementation are discussed in the following section

3.2 Overview of selected algorithms

3.2.1 Linear regression

Regression problems are supervised learning problems in which the response is continuous. Linear regression is a technique that is used for regression problems. Simple linear regression is an approach for predicting a **quantitative response** using a single feature (or "predictor" or "input variable"). It takes the following form:

$$y = \beta_0 + x\beta_1$$

Here y is the response, x is the feature, β_0 is the intercept, β_1 is the coefficient for x. Together, β_0 and β_1 are called the **model coefficients**. Coefficients are estimated using the **least squares criterion**, which means we find the line (mathematically) which minimizes the **sum of squared residuals** (or "sum of squared errors"):

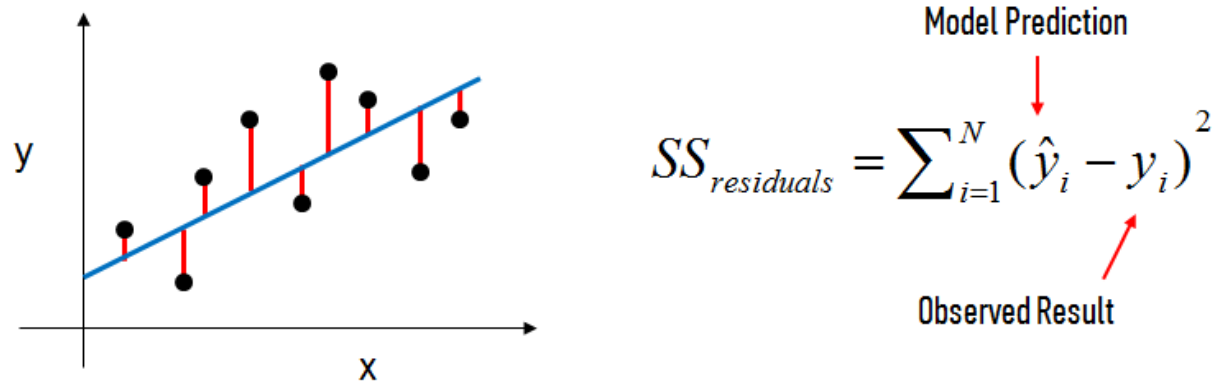


Figure 3.1: Linear regression Procedure

In this figure, the black dots are the observed values of x, y. The blue line is our least squares line. The red lines are the residuals, which are the distances between the observed values and the least squares line.

The model coefficients relate to the least squares line $\boldsymbol{\beta_0}$ is the intercept (the value of y when x=0) and $\boldsymbol{\beta_1}$ is the slope (the change in y divided by change in x). Here is a graphical representation of those calculations:

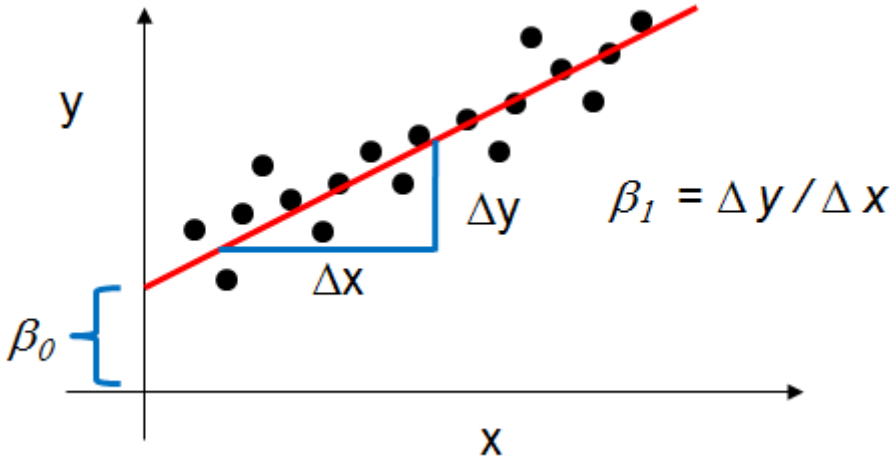


Figure 3.2: Co-efficient calculation of linear regression

Multiple Linear Regression:

Simple linear regression can easily be extended to include multiple features. This is called **Multiple Linear Regression**:

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

Each *x* represents a different feature and each feature has its own coefficient. From our datasets if we want to predict the profit of a product and if we use 'Sales', 'Quantity' and 'Discount' as a feature then the equation will be looked like this:

$$Profit = \beta_0 + \beta_1 \times Sales + \beta_2 \times Quantity + \beta_3 \times Discount$$

Model Evaluation Metrics for Regression:

The metrics that are popular for regression problems are Mean Absolute Error, Mean Squared Error and Root Mean Squared Error.

a. **Mean Absolute Error (MAE)** is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - py_i|$$

b. **Mean Squared Error (MSE)** is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - py_i)^2$$

c. **Root Mean Squared Error (RMSE)** is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - py_i)^2}$$

Here, y_i is the original value of target variable and py_i is the predicted value of target variable and n is the number of samples. **MSE** is more popular than **MAE** because MSE "punishes" larger errors. But, **RMSE** is even more popular than MSE because **RMSE** is interpretable in the "y" units. We will **use the RMSE** metrics in our implementation.

3.2.2 Polynomial Regression

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modeled as an n^{th} degree polynomial in x. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_m x_i^m (i = 1,2,3 \ldots, n)$$

The goal of regression analysis is to model the expected value of a dependent variable *y* in terms of the value of an independent variable *x*. A second order (m=2) polynomial forms a quadratic expression (parabolic curve), a third order (m=3) polynomial forms a cubic expression and a fourth order (m=4) polynomial forms a quartic expression. In many settings, such a linear relationship may not hold. That's where the polynomial regression become popular. Some important point regarding polynomial regression is:

- The fitted model is more reliable when it is built on large numbers of observations.
- Do not extrapolate beyond the limits of observed values.
- Choose values for the predictor (x) that are not too large as they will cause an overflow with higher degree polynomials; scale x down if necessary.

3.2.3 Logistic Regression

It is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a logit transformation of the probability of the presence of the characteristic of interest.

Let assign 1 if an email is spam and 0 if it's not. So, our prediction choices can be written as:

$$P(Y=1|x;\theta) \text{ and } P(Y=0|x;\theta)$$

Then we'll chose a threshold value for our prediction function $\mathbf{h_\theta(x)}$ is 0.5. If $\mathbf{h_\theta(x) \geq 0.5}$ then Y = 1 and if $\mathbf{h_\theta(x) < 0.5}$ then Y = 0;

Now, if we have the training set,

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots\ldots, (x^{(m)}, y^{(m)}),\}$$

For m examples $x = \begin{pmatrix} x0 \\ x1 \\ x2 \\ \ldots. \\ xn \end{pmatrix}$ where, $x_0 = 1$ and Y € {0,1}

We can write our hypothesis of the logistic function as $\mathbf{h_\theta(x) = 1 /1+ e^{-\theta x}}$**.** This function is called **Sigmoid function** or logistic function. Now we have to choose the parameter θ to fit the function.

Cost Function: we can denote the cost function as j**,**

So, $j(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{cost}(h_\theta(x^{(i)}), y^{(i)})$

$= \frac{1}{m} [\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1- y^{(i)}) \log(1- h_\theta(x^{(i)}))]$, m is the number of total examples.

To fit the parameter θ we have to take the minimum of j(θ).

Gradient Descent: from the previous section, we know the cost function j(*θ):*

$J(\theta) = \frac{1}{m} [\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1- y^{(i)}) \log(1- h_\theta(x^{(i)}))]$

Now we will minimize the cost function by repeating the following formula:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta} j(\theta)$$

$$\theta_j = \theta_j - \alpha \sum_{-=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

where α is the learning rate of the equation.

Finally, if we update the value of minimized θ_j to the hypothesis, we will get the logistic regression equation.

3.2.4 K-Nearest Neighbor

K-Nearest Neighbor (in short KNN) is a supervised learning algorithm. This means a labeled dataset is given consisting of training observations (x, y) and we would like to capture the relationship between x, y. More formally, our goal is to learn a function h: X→Y, so that given an unseen observation x, h(x) can confidently predict the corresponding output y. KNN is a non-parametric and instance-based algorithm.

Non-Parametric means it makes no explicit assumptions about the functional form of h, avoiding the dangers of miss-modeling the underlying distribution of the data. For example, suppose our data is highly non-Gaussian but the learning model we choose assumes a Gaussian form. In that case, our algorithm would make extremely poor predictions.

Instance-Based learning means that our algorithm doesn't explicitly learn a model. Instead, it chooses to memorize the training instances which are subsequently used as "knowledge" for the prediction phase.

The K-nearest-neighbor (KNN) algorithm measures the distance between a query scenario and a set of scenarios in the data set. We can compute the distance between two scenarios using some distance function d(x, y), where x, y are scenarios composed of K features, such that:

X = {x1, …., xN}, Y = {Y1, ……, YN}.

Two distance measuring techniques are given below:

Euclidian Distance: $\mathbf{d(X,Y) = \sum_{i=1}^{K} \sqrt{xi2 - yi2}}$

Manhattan Distance: $\mathbf{d(X,Y) = \sum_{i=1}^{K} |X1 - Y1|}$

Now that we have established a measure in which to determine the distance between two scenarios, we can simply pass through the data set, one scenario at a time and compare it to the query scenario.

We can represent our data set as a matrix M = N x P , containing P data points D1,……,DP, where each datapoint contains N features Di = { Di1,……, Dnp }. A vector ***v*** with length P of output values **v = {v0,……vp}** accompanies this matrix, listing the output value **vi** for each data point Di .

It should be noted that the vector ***v*** can also be a column matrix; if multiple output values are desired, the width of the matrix may be expanded.

Chapter 4
Designing of Data Warehouse for Mining in Sales Data

4.1 Initial transactional Data

The first dataset, we decided to work is an Open source dataset and collected from Tableau Community [13]. This dataset has 24 Columns, 51290 transactional tuples, across 147 countries. There are 10768 products in our dataset which divided into three categories and 17 Subcategories. We will apply data warehouse for this dataset only. We have used another dataset which is also collected from Tableau community [20]. Our second dataset contains 9426 tuples, 24 attributes. There are 1263 products divided into three categories. Throughout the whole report, we will use these two datasets and will address them as our **'First Dataset'** and **'Second Dataset'**.

4.2 Schema of Data Warehouse

The most popular data model for a data warehouse is a multidimensional model which can exist in the form of a star schema, a snowflake schema, or a fact constellation schema.

We have designed our Data Warehouse using snowflake schema. The model, we have designed, has fourteen-dimension tables and one fact table. The fact table contains 15 attributes in which one is used as a primary key, 8 attributes are used as a foreign key for the dimension tables and rest of the attributes are different measures of a single tuple.

There are some dimension tables which contain some new attributes out of our dataset. The Order Date and Ship Date both dimension tables have detailed information about time. In our dataset, we only have a date. For mining purpose, we now break this date into the day, month, year and quarter. Similarly, for any particular product, we did not have the actual price and selling price. We also find out that from our existing data. These new attributes will give us a better understanding of our data. The design of data warehouse is shown in figure 4.1.

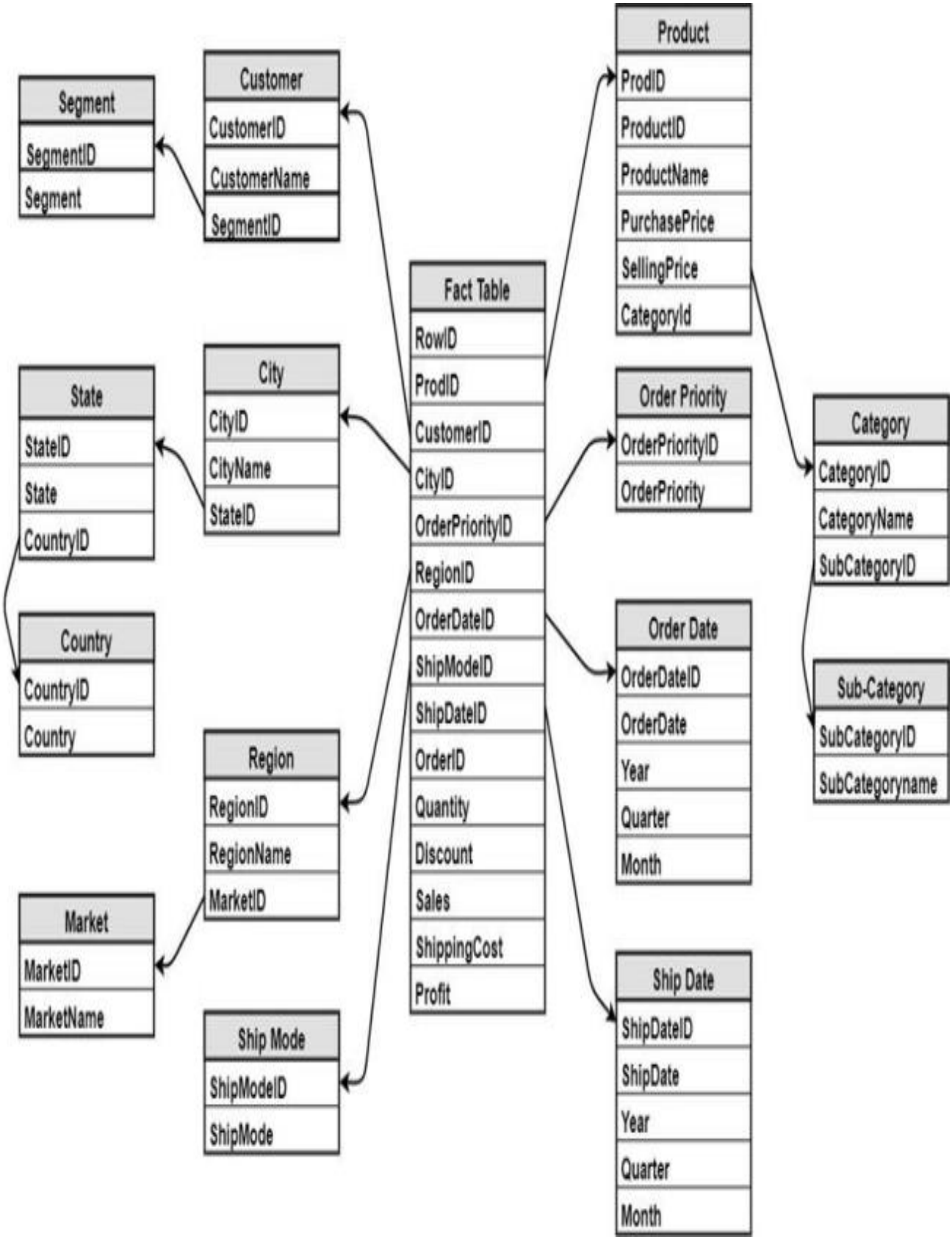


Figure 4.1: Snow-flake schema of our Data Warehouse

4.3 Data Dictionary of Data Warehouse

A data dictionary is a list of definitions used in the system. Each definition is usually 10 to 50 words long and there are about 50 to 200 definitions. Which are mostly technical terms, such as the meaning of each field in the database. The data dictionary of our data warehouse is given below.

Column Name	DATA TYPE	Constraints	Description
RowID	VARCHAR(50 BYTE)	Primary Key	Fact Table Row ID
ProdID	VARCHAR(50 BYTE)	Foreign Key	ProdID from Product dimension table
CustomerID	VARCHAR(50 BYTE)	Foreign Key	CustomerID from Customer dimension table
CityID	VARCHAR(50 BYTE)	Foreign Key	City ID from City dimension table
OrderPriorityID	VARCHAR(50 BYTE)	Foreign Key	OrderPriorityID from OrderPriority dimension table
RegionID	VARCHAR(50 BYTE)	Foreign Key	RegionID from region dimension table
OrderDateID	VARCHAR(50 BYTE)	Foreign Key	OrderDateID from OrderDate dimension table
ShipModeID	VARCHAR(50 BYTE)	Foreign Key	ShipModeID from ShipMode dimension table
ShipDateID	VARCHAR(50 BYTE)	Foreign Key	ShipDateID from Shipdate dimension table
OrderID	VARCHAR(50 BYTE)	-	OrderID of products
Quantity	NUMBER(7,0)	-	Quantity of products
Sales	NUMBER(7,4)	-	Transaction amount
Discount	NUMBER(7,4)	-	Discount on products
ShippingCost	NUMBER(7,4)	-	Cost of shipping
Profit	NUMBER(7,4)	-	Benefits of selling

Table 4.1: Fact table of our data warehouse

Column Name	DATA TYPE	Constraints	Description
CustomerID	VARCHAR(50 BYTE)	Primary Key	Customer ID
CustomerName	VARCHAR(50 BYTE)	-	Name of the customer
SegmentID	VARCHAR(50 BYTE)	Foreign Key	Segment ID

Table 4.2: Dimension Table of Customer

Column Name	DATA TYPE	Constraints	Description
CityID	VARCHAR(50 BYTE)	Primary Key	City ID
CityName	VARCHAR(50 BYTE)	-	City Name
StateId	VARCHAR(50 BYTE)	Foreign Key	State ID

Table 4.3: Dimension Table of City

Column Name	DATA TYPE	Constraints	Description
RegionID	VARCHAR(50 BYTE)	Primary Key	Region ID
RegionName	VARCHAR(50 BYTE)	-	Region Name
MarketID	VARCHAR(50 BYTE)	Foreign Key	Market ID

Table 4.4: Dimension Table of Region

Column Name	DATA TYPE	Constraints	Description
OrderPriorityID	VARCHAR(50 BYTE)	Primary Key	Unique Order Priority ID
OrderPriority	VARCHAR(50 BYTE)	-	Priority of ordered product

Table 4.5: Dimension Table of Order Priority

Column Name	DATA TYPE	Constraints	Description
CountrytID	VARCHAR(50 BYTE)	Primary Key	Unique Country ID
Country	VARCHAR(50 BYTE)	-	Name of the country

Table 4.6: Dimension Table of Country

Column Name	DATA TYPE	Constraints	Description
ShipModeID	VARCHAR(50 BYTE)	Primary Key	Unique Ship Mode ID
ShipMode	VARCHAR(50 BYTE)	-	Mode of shipping

Table 4.7: Dimension Table of Shipping Mode

Column Name	DATA TYPE	Constraints	Description
ProdID	VARCHAR(50 BYTE)	Primary Key	Unique key defining unique product id and name
ProductID	VARCHAR(50 BYTE)	-	Product ID
ProductName	VARCHAR(50 BYTE)	-	Product Name
PurchasePriceUnit	NUMBER(7,4)	-	Price of purchasing a product
SellingPricePerUnit	NUMBER(7,4)	-	Price of selling a product
CategoryID	VARCHAR(50 BYTE)	Foreign Key	Category ID of a product

Table 4.8: Dimension Table of Product

Column Name	DATA TYPE	Constraints	Description
OrderDateID	VARCHAR(50 BYTE)	Primary Key	Unique Order Date ID
OrderDate	DATE	-	Date of order
Year	NUMBER(7,0)	-	Year portion of order date
Quarter	VARCHAR(50 BYTE)	-	A quarter of the year
Month	VARCHAR(50 BYTE)	-	The month of order date

Table 4.9: Dimension Table of Order Date

Column Name	DATA TYPE	Constraints	Description
ShipDateId	VARCHAR(50 BYTE)	Primary Key	Unique Ship Date ID
ShipDate	DATE	-	Date of shipping
Year	NUMBER(7,0)	-	Year portion of the shipping date
Quarter	VARCHAR(50 BYTE)	-	A quarter of shipping date
Month	VARCHAR(50 BYTE)	-	The month of the shipping date

Table 4.10: Dimension Table of Shipping Date

Column Name	DATA TYPE	Constraints	Description
SegmentID	VARCHAR(50 BYTE)	Primary Key	Unique Segment ID
Segment	VARCHAR(50 BYTE)	-	Name of the segment

Table 4.11: Dimension Table of Segment

Column Name	DATA TYPE	Constraints	Description
StatetID	VARCHAR(50 BYTE)	Primary Key	Unique State ID
State	VARCHAR(50 BYTE)	-	Name of the state
CountryID	VARCHAR(50 BYTE)	Foreign Key	Country ID from the country dimension table

Table 4.12: Dimension Table of State

Column Name	DATA TYPE	Constraints	Description
MarketID	VARCHAR(50 BYTE)	Primary Key	Unique Market ID
MarketName	VARCHAR(50 BYTE)	-	Name of the market

Table 4.13: Dimension Table of Market

Column Name	DATA TYPE	Constraints	Description
CategoryID	VARCHAR(50 BYTE)	Primary Key	Unique Category ID
CategoryName	VARCHAR(50 BYTE)	-	Name of the category
SubCategoryID	VARCHAR(50 BYTE)	Foreign Key	SubCategory ID from SubCategory Dimension table

Table 4.14: Dimension Table of Category

Column Name	DATA TYPE	Constraints	Description
SubCategorytID	VARCHAR(50 BYTE)	Primary Key	Unique Sub-Category ID
SubCategory	VARCHAR(50 BYTE)	-	Name of the sub-category

Table 4.15: Dimension Table of Sub Category

Chapter 5
Implementation of Data Mining Algorithms

5.1 Feature selection and model evaluation technique

Feature selection is useful as a preprocessing step to improve computational efficiency in predictive modeling. Oracle Data Mining implements feature selection for optimization. We use **Pearson Correlation** in our research for feature selection for regression problem and **Recursive Feature Elimination with Cross Validation (RFECV)** for classification problem. To evaluate the predictive models we will use **K-fold cross-validation.**

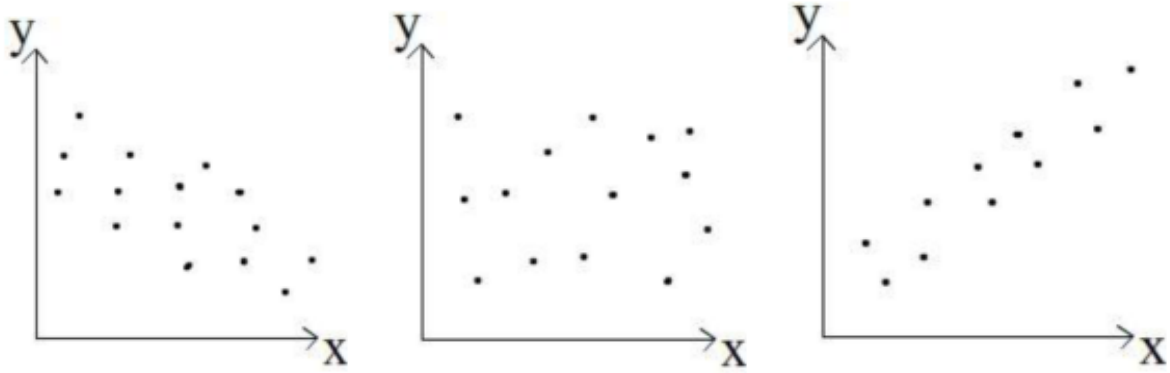
5.1.1 Pearson Correlation

Correlation between sets of data is a measure of how well they are related. The most common measure of correlation in stats is the Pearson Correlation. It shows the linear relationship between two sets of data. In simple terms, it answers the question, Can I draw a line graph to represent the data?

We can categorize the type of correlation by considering as one variable increases what happens to the other variable:

- **Positive correlation** – the other variable has a tendency to increase;
- **Negative correlation** – the other variable has a tendency to decrease;
- **No correlation** – the other variable does not tend to either increase or decrease.

The starting point of any such analysis should thus be the construction and subsequent examination of a **scatterplot**. Examples of negative, no and positive correlation are as follows.



Negative Correlation No Correlation Positive Correlation

Figure 5.1: Correlation

Pearson’s correlation coefficient is a statistical measure of the strength of a linear relationship between paired data. In a sample it is denoted by r and is by design constrained as follows:

- Positive values denote positive linear correlation;
- Negative values denote negative linear correlation;
- A value of 0 denotes no linear correlation;
- The closer the value is to 1 or –1, the stronger the linear correlation.

The general formula of Pearson’s coefficient is:

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2 \,][\, n\Sigma y^2 - (\Sigma y)^2 \,]}}$$

Where x is the feature and y is the dependent variable or target variable and n is the number of samples. In Python, we have a built-in library to calculate Pearson coefficient.

5.1.2 Recursive Feature Elimination with Cross-Validation

For classification with small training samples and high dimensionality, feature selection plays an important role in avoiding overfitting problems and improving classification performance. One of the commonly used feature selection methods for small samples problems is **Recursive Feature Elimination (RFE)** method. RFE method utilizes the generalization capability embedded in support vector machines and is thus suitable for small samples problems. Despite its good performance, RFE tends to discard "weak" features, which may provide a significant improvement of performance when combined with other features. We initially start with all the features. For every step or iteration, the worst x number of features are eliminated using the "step" parameter till "n-features" are left. If you notice, you need to provide the n-features parameter in the constructor.

The RFECV object helps to tune or find this n_features parameter using cross-validation. For every step where "step" number of features are eliminated, it calculates the score on the validation data. The number of features left at the step which gives the maximum score on the validation data is considered as "the best n_features" of data. To use RFECV, we

need to import RFECV library from sklearn.feature_selection. The RFECV function has some parameters.

*Class sklearn.feature_selection.**RFECV***(estimator, *step=1,cv=None, scoring=None,verbose=0, n_jobs=1*)

Here, **Estimator:** a supervised learning estimator, **Step** is the number of features eliminate at each iteration, **CV** Determines the cross-validation splitting strategy and Scoring is a string or a scorer callable object/function with signature (estimator, X, y). Rest of the parameters are not important.

After creating an instance of RFECV we call the Fit function with our feature vector and the target variable. The RFECV model will automatically tune the number of selected features. RFECV has two important attributes named **n_features_, ranking_**. **n_features_** returns the number of selected features with cross-validation. **ranking_** is an array that returns **t**he feature ranking, such that **ranking_[i]** corresponds to the ranking position of the i^{th} feature. Ranking with 1 indicates the best feature.

5.1.3 K-fold Cross-Validation

In train-test split, we split the dataset into two pieces, so that the model can be trained and tested on different data. Testing accuracy is a better estimate than training accuracy of out-of-sample performance. But, it provides a high variance estimate since changing which observations happen to be in the testing set can significantly change testing accuracy. What if we created a bunch of train/test splits, calculated the testing accuracy for each and averaged the results together? That's the essence of cross-validation.

Steps for K-fold cross-validation:

1. Split the dataset into K **equal** partitions (or "folds").
2. Use fold 1 as the **testing set** and the union of the other folds as the **training set.**
3. Calculate **testing accuracy**.
4. Repeat steps 2 and 3 K times, using a **different fold** as the testing set each time.
5. Use the **average testing accuracy** as the estimate of out-of-sample accuracy.

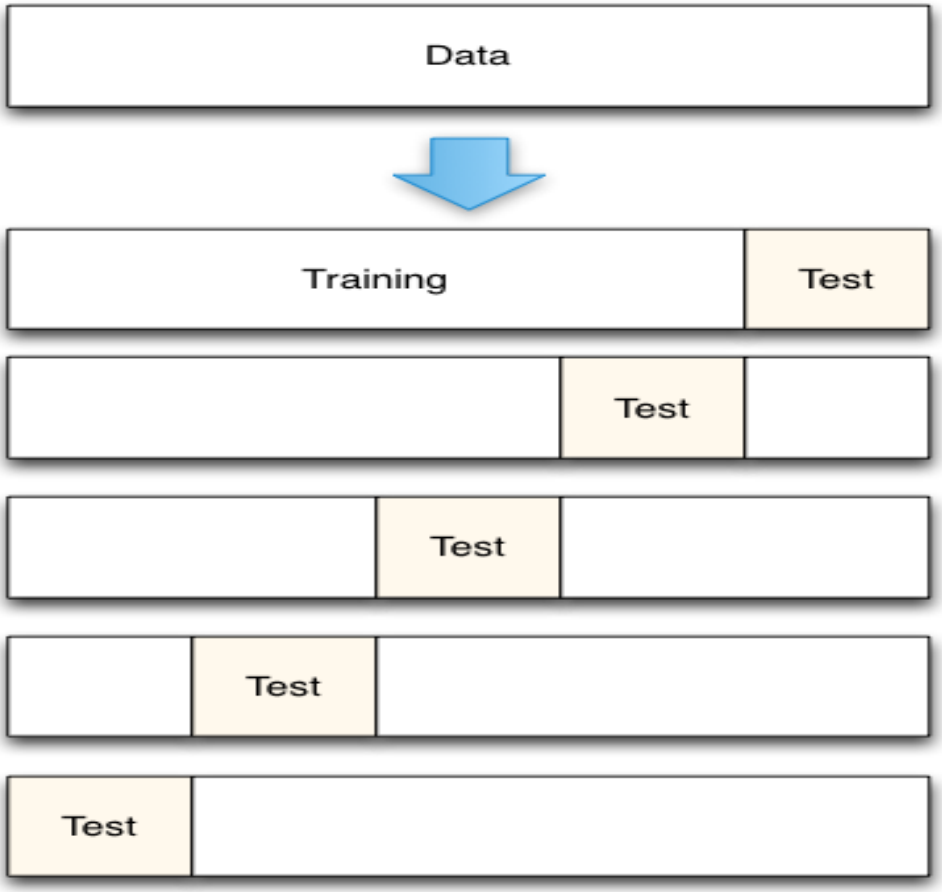


Figure 5.2: 5-fold cross-validation

Comparing cross-validation to train/test split.

Advantages of **cross-validation**:

- More accurate estimate of out-of-sample accuracy
- More "efficient" use of data (every observation is used for both training and testing).

5.2 Implementation of Algorithms

5.2.1 Linear Regression Implementation

The first problem we want to solve is predicting the profit of a particular product using linear regression algorithm. After analyzing our first dataset, we have seen that the highest selling product is “Binney & Smith Sketch Pad, Blue” which product id is “OFF-BIN-10002061”. This product has four variances but the “OFF-BIN-10002061” has been sold 23 times. Our goal is to predict the profit of this product from our dataset.

According to Pearson Correlation, we have selected our features. We have seen that 'Sales', 'Discount' and 'Quantity' has a linear relationship with our target variable 'Profit'. We have selected 'Sales', 'Discount' and 'Quantity' as our features vector. Our target variable will be 'Profit'.

We will use cross-validation to evaluate the predictive model. **Root Mean Squared Error (RMSE)** metrics have been used. Since we have a little amount of data for a single product, we will use 3-fold cross validation then calculated the average RMSE value. We will also use train-test split and observe the difference between cross-validation and train-test split. We get an RMSE value of 5.96993034214e-14 which is near to zero. In train-test split we got 6.29793031992e-14.

Now we want to predict the profit of any product using linear regression. After analyzing our first dataset, we have seen that we have nearly 10768 products. Now we will try to build a model that can predict any product's profit. Here we will use full dataset that means all 51290 tuples to build our model.

According to the previous problem, first, we will select some important features using Pearson correlation. But here we see that there is no strong relationship between any features with the target variable. All Pearson coefficient value is between -.316 to .485**.** So, it is clear to us that there is no linear relationship exist and it will give us high RMSE value if we want to apply linear regression. Now we have to pick some features that are not very close to zero and use them as our input features. We will select 'SellingPriceperunit', 'PurchasingPriceperUnit', 'Sales', 'Quantity' and 'Discount' as our feature vector and 'Profit' as our target variable.

Here we have plenty of data so we have used 10-fold cross-validation. We got the RMSE value of 98.76.

We also have another attribute named shipping cost which indicates the cost required to ship a product. Using linear regression we can also predict shipping cost. The working procedure of linear regression is known to us now. Like before, first we try to build a model which predict the shipping cost of a particular product. Then we will attempt to build a model that can predict shipping cost for any product. One thing should be mentioned that

previously we have used Pearson correlation to take features. It gives us a better understanding of which attribute has a linear relation with target variable but it also has limitations. Combination of non-linear features can be used as a good predictor for the model.

"Binney & Smith Sketch Pad, Blue" is the selected product for which we want to predict the shipping cost. So we separate this product data from our first dataset and train our model using this data. The input features are 'Sales', 'Quantity', 'OrderpriorityID', 'ShipmodeID' and 'RegionID'. Target variable is 'ShippingCost'. Here we have a little amount of data so we have used 3-fold cross-validation. We got the RMSE value of 3.31. We did not take the features according to Pearson correlation. If we take features according to Pearson Co-efficient, then we get a higher RMSE value. In the first case, we took 'RegionID' as our input feature which has no linear relationship with the target variable but still gave us lower RMSE. But when we take the feature that has a strong linear relation with the target variable we have higher RMSE.

We have got reasonable RMSE while predicting shipping cost of a particular product. Now we try to predict the shipping cost of all products. After observing the values of Pearson Correlation, we see that there is no strong relationship with 'ShippingCost' with any attribute. Only Sales have a slightly strong relationship with shipping cost. So, we took only Sales as our input feature. The RMSE value we get after evaluating the model is 29.26.

Linear Regression implementation on our second dataset:

Now we will try these algorithms on the different dataset to perceive the information how these algorithms work on another dataset. In our second dataset we will first predict the profit of a single product, then we will try to predict the profit of any product. After that using the same way, we will predict the shipping cost for a particular product and then the shipping cost of any product.

"**Global High-Back Leather Tilter, Burgundy"** is the selected product for which we want to predict the profit. So we separate this product data from our dataset and train our model using these data. The input features are 'Sales', 'Quantity ordered new' and 'Discount'. Target variable is Profit. Here we have 27 samples so we have used 3-fold cross-validation. We got the RMSE of 800.21 which is a very high error rate. We took the features

accordingly Pearson correlation. But still, it gave us higher RMSE. Now we try to predict the shipping cost of a single product. The input features are 'Unit Price' and 'Product Base Margin'. Target variable is 'Shipping Cost'. Here we got 0.0 RMSE value. It means our model predicts the shipping accurately without any error. Now we want to predict the profit of any product using linear regression. After analyzing our second dataset, we see that we have nearly 1304 products. Now we will try to build a model that can predict any product's profit. There are some rows in which where we have incomplete data. After removing them from our second dataset, we got 9354 tuples to build our model.

Now we want to predict the profit of any product using linear regression. After analyzing our second dataset, we see that we have nearly 1304 products. Now we will try to build a model that can predict any product's profit. Here we will use the full dataset that means all 9354 tuples to build our model.

We will select some important features using correlation matrix. The input features are 'Sales' and 'Quantity ordered new'. Target variable is 'Profit'. Here we are using full data set, so we have used 10-fold cross-validation. We got the RMSE value of 913.08. Similarly, now we will try to predict the 'Shipping Cost'. The input features are 'Sales' and 'Product Base Margin'. Target variable is 'Shipping Cost''. Here the RMSE for this model is 15.22.

5.2.2 Polynomial Regression Implementation

Previously, we did not get a good model for predicting the profit of any product using linear regression. In this section, we have tried to predict the profit and shipping cost using polynomial regression. First of all, we have selected 'SellingPriceperunit', 'PurchasingPriceperUnit', 'Sales', 'Quantity' & 'Discount' as our feature vector and 'Profit 'as our target variable. We have used cross-validation to evaluate the model. RMSE metrics have been used. Here we have plenty of data so we have used 10-fold cross validation then calculated the average RMSE value. We run polynomial regression using degree 1 to 4. We get **0.34** RMSE value which is near to zero using polynomial with **Degree 3**.

Using linear regression, we also get higher RMSE value in predicting shipping cost. Now we will to use polynomial regression and observe our model's performance improves or not. "Binney & Smith Sketch Pad, Blue" is the selected product for which we want to

predict the shipping cost. So we separate this product's data from our dataset and train our model using these data. The input features are Sales, Quantity, OrderpriorityID, ShipmodeID & RegionID. Target variable is Shipping Cost. Here we have a little amount of data so we have used 3-fold cross-validation. We got the RMSE value of 3.31 when we use polynomial with degree 1. We got higher RMSE value 23.18, 261.58 for using polynomial regression with degree 2, 3 respectively.

When we take all products data and try to predict shipping cost of any product using polynomial regression using degree 1 to 5 we get the RMSE value of 29.26, 26.27, 26.74, 30.32 and 89.53 respectively.

Polynomial regression implementation on our second dataset:

Previously, we did not get a good model for predicting the profit of a single product using linear regression in our second dataset. Now we will try to predict the profit and shipping cost using polynomial regression. We have selected 'Sales', 'Quantity ordered new' and 'Discount' as our feature vector and 'Profit' as our target variable. We used cross-validation to evaluate the model. RMSE metrics also has been used here like before. The mean RMSE for polynomial regression using degree 2 to 4 are 8132.20, 6665771.72 & 8154381.86 respectively. For predicting the 'Shipping Cost' of a single product we got the RMSE value of 0.0 in all cases.

When we take all products data and try to predict 'Profit' using polynomial regression using degree 2 to 4, we have got the RMSE value of 924.98, 961.37 and 2313.88 respectively. In 'Shipping Cost' prediction using polynomial regression using degree 2 to 4, we have got the RMSE value of 15.41, 21.82 and 48.61 respectively.

5.2.3 Logistic Regression Implementation

In this section, our goal is to detect the order priority of any product using Logistic regression. Since Order Priority has discrete value, we are using classification algorithm. To prepare the dataset, first, we need to identify the important features. In case of classification problem, we have used **Recursive feature elimination with cross-validation** (RFECV). For predicting 'OrderPriorityID', we have checked the ranking of all attributes using RFECV. Then we have selected the feature which has rank 1. After running RFECV we have got 3 features ranking 1. They are 'DateDif', 'Discount' and

‘CategoryID’. So, we have used these three features as our input feature. Our target variable is ‘OrderPriorityID’. We used 10 fold cross-validation to evaluate the model. Accuracy has been used as a scoring parameter. Despite selecting the best features, we got 62.73% accuracy.

We also have another attribute called ‘ShipModeID’ which also contains discrete values and an important feature in our dataset. To predict the ship mode of any product first we need to identify the important features. For predicting the ‘ShipModeID’, we have checked the ranking of all attributes using RFECV. Then we have selected the feature which has rank 1. After running RFECV we get one feature as ranking one which is ‘DateDif’. So, we take this as an input feature and evaluate our model using 10 fold cross-validation. It gives us 78.11% accuracy.

5.2.4 K-Nearest Neighbor Implementation

In this section, our goal is to predict the order priority and ship mode using K-nearest neighbor. Like logistic regression, to prepare the dataset first we need to identify the important features. For predicting ‘OrderPriorityID’, we have selected ‘DateDif’, ‘Discount’ and ‘CategoryID’ as our input features. We will run KNN for k= 1 to 700 and select the best k to build our model. We have achieved 61.57% accuracy.

Using same techniques, we have tried to predict the ‘ShipModeID’ of any product. This time we have selected ‘DateDif’ as our input feature and run KNN for k=1 to 60. The reason for selecting the range of k differently for two will be discussed in result analysis section. Here we have achieved 81.40% accuracy.

5.3 Improving Regression Model

In the Previous section, we have seen that we did not get any good model for predicting the ‘ShippingCost’ of any product in our first dataset. In our second dataset, we have also faced the same problem while we try to predict the ‘Profit’ and ‘Shipping Cost’. In the following section, we will try to improve these models’ performance.

5.3.1 Variable Transformation

Data manipulation is an important part of the data mining process. In order to ensure that all the data is used effectively, it can sometimes be used to transform the variables. Log transformation is the most popular one for right-skewed distributions in linear regression. Logarithms of all-positive variables because this leads to multiplicative models on the original scale, which often makes sense. Constant ratios tend to have constant differences. This makes logs relatively easy to interpret since constant percentage changes become a constant shift. So a decrease of the natural log is a 15% decrease in the original numbers, no matter how big the original number is.

A lot of economic and financial data behaves like this, for example, constant or near-constant effects on the percentage scale. The log scale makes a lot of sense in that case, as a result of that percentage-scale effect. The spread of values tends to be larger as the mean increases and taking logs also tends to stabilize the spread. That is usually more important than normality. If your data are log-normally distributed, then the log transformation makes them normally distributed. Normally distributed data have lots going for them.

The normal distribution is widely used in basic and clinical research studies to model continuous outcomes. Unfortunately, the symmetric bell-shaped distribution often does not adequately describe the observed data from research projects. Quite often data arising in real studies are so skewed that standard statistical analyses of these data yield invalid results. Many methods have been developed to test the normality assumption of observed data. When the distribution of the continuous data is non-normal, transformations of data are applied to make the data as "normal" as possible and thus, increase the validity of the associated statistical analyses. The log transformation is, arguably, the most popular among the different types of transformations used to transform skewed data to approximately conform to normality.

If the original data follows a log-normal distribution or approximately so, then the log-transformed data follows a normal or near normal distribution. In this case, the log-transformation removes or reduces skewness. Unfortunately, data arising from many studies do not approximate the log-normal distribution so applying this transformation does not reduce the skewness of the distribution. In fact, in some cases applying the transformation can make the distribution more skewed than the original data.

In general, for right-skewed data, the log-transformation may make it either right-or left-skewed. If the original data follows a log-normal distribution, the log-transformed data will follow or approximately follow the normal distribution. However, in general, there is no guarantee that the log-transformation will reduce skewness and make the data a better approximation of the normal distribution.

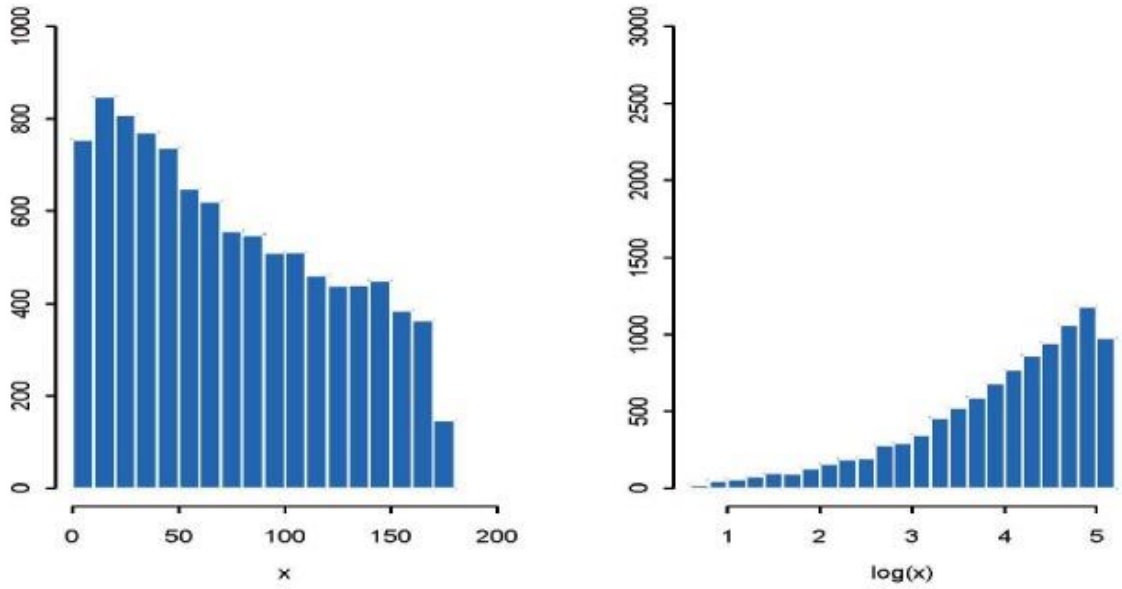


Figure 5.3: Logarithmic transformation of some data

Another popular use of the log transformation is to reduce the variability of data, especially in data sets that include outlying observations. Again, contrary to this popular belief, log transformation can often increase the variability of data whether there are outliers. Apart from transformations, creating new variables out of existing variables is also very helpful.

Using Variable Transformation in Our First Dataset:

To improve the performance of regression classifier, we have transformed some attributes into their natural logarithmic value and use them to train our model. In our first dataset, we have transformed the ‘Sales’, ‘ShippingCost’, ‘Profit’ and ‘Quantity’ attributes into their natural logarithm values and have created new attributes named ‘Log_Sales’, ‘Log_ShippingCost’, ‘Log_Profit’ and ‘Log_Quantity’ respectively. Then we have generated the correlation matrix by taking some correlated features and have shown it using heat map. The heat map is shown in Figure 5.4.

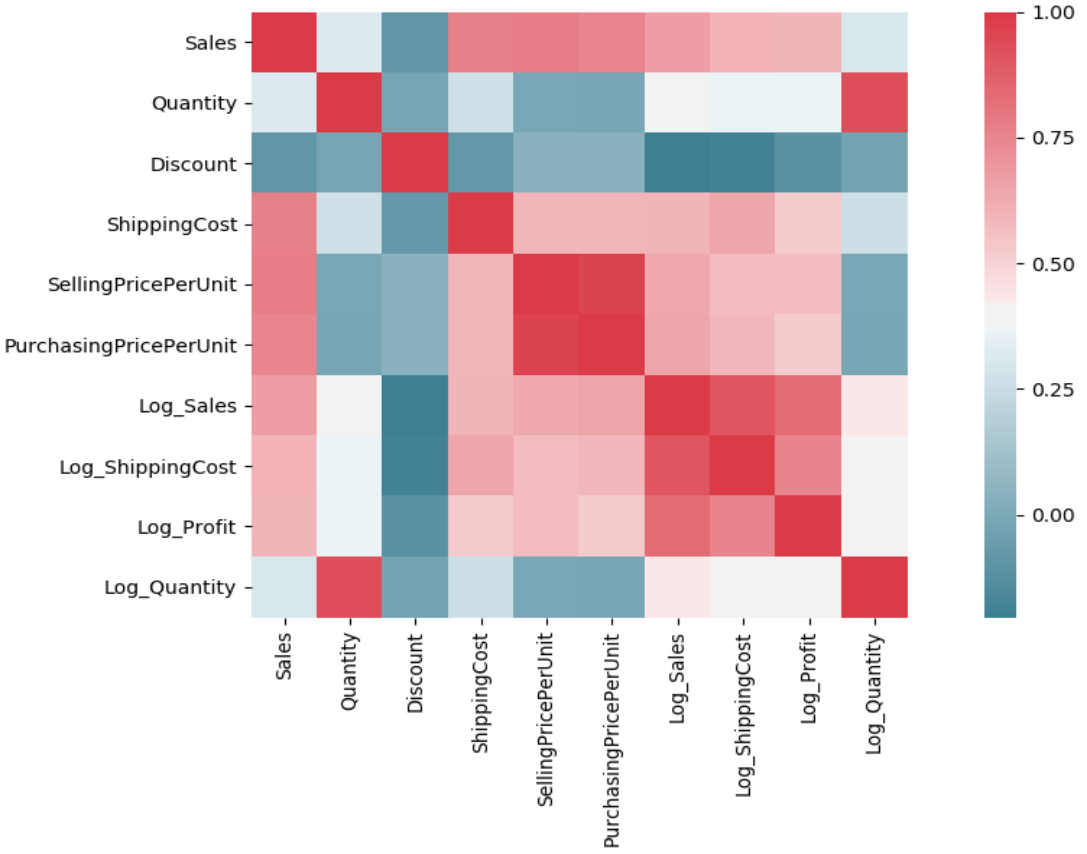


Figure 5.4: Correlation matrix of the first dataset using heat map

From the heat map of the first dataset, we see that ‘ShippingCost’ has correlated with ‘Log_Sales’, ‘Log_Profit’ and ‘Quantity’. Since we will use polynomial regression, we can add some more features to improve the performance. We will select 'Log_Sales', 'Quantity', 'OrderPriorityID', 'ShipModeID' and 'RegionID' as our feature vector and ‘ShippingCost’ as our target variable. The mean RMSE for polynomial regression using degree 1 to 4 are 39.46, 36.01, 30.97 and 30.71 respectively.

One thing should be mention here that ‘Log_Profit’ also shows a strong correlation with the target variable, but we did not take it as our input feature. The reason is profit can contain negative values. So when we take the natural logarithm of a negative value it will be undefined. So ‘Log_Profit’ will contain many empty values where the profit is negative. To train our model, we have to remove those rows. Here we will not be able to predict the shipping cost of a product when the profit of that product is negative. So we did not take ‘Log_Profit’ as our input feature.

Using Variable Transformation in Our Second Dataset:

We move on to our second dataset. In our second dataset we also have transformed the ‘Sales’, ‘Shipping Cost’, ‘Quantity ordered new’ and ‘Profit’ attributes into their natural logarithm values and have created new attributes named ‘Log_Sales’, ‘Log_ShippingCost’, ‘Log_Quantity’ and ‘Log_Profit’ respectively. Here we also have converted ‘Ordered quantity new’ because the value of these attributes varies a lot. We have generated the correlation matrix by taking some features and have shown it using heat map. The heat map is shown in figure 5.5.

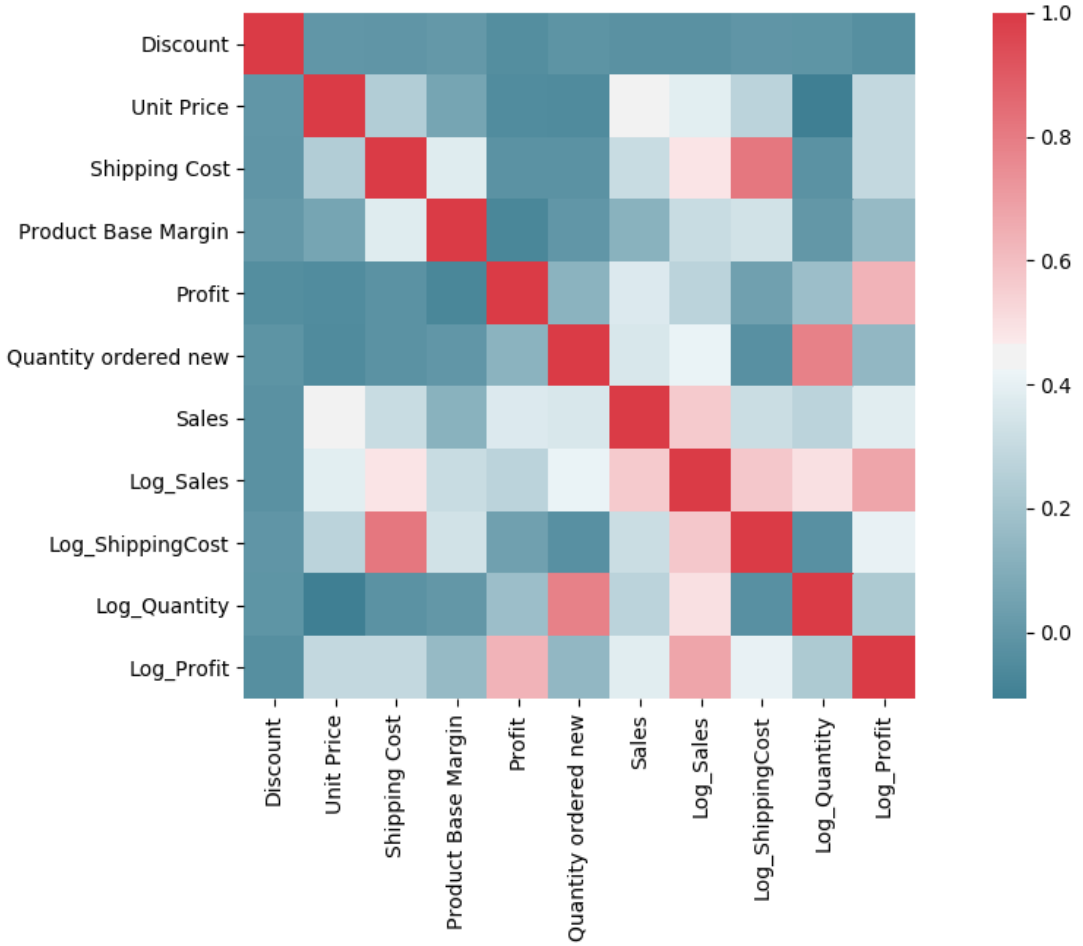


Figure 5.5: Correlation matrix of first dataset using heat map

Previously we have seen that while predicting the profit of a product we have higher value of RMSE. Now from the heat map, we observe that if we want to predict the 'Profit' we do not have any good feature. We have selected 'Log_Sales', 'Log_ShippingCost' and 'Log_Quantity' as our feature vector and 'Profit' as our target variable. The mean RMSE for polynomial regression using degree 1 to 4 are 931.87, 854.33, 778.98 and 767.42 respectively.

Now we will try to predict the 'Shipping Cost' from our second dataset. From the heat map, we see 'Shipping Cost' strongly correlated with 'Log_Sales' and 'Product Base Margin'. We have selected these two attributes as our feature vector and 'Shipping Cost' as our target variable. The mean RMSE for polynomial regression using degree 1 to 4 is 14.37, 13.84, 13.54 and 13.34 respectively.

5.3.2 Outlier Analysis

Outliers is a data object that deviates significantly from the rest of the objects as if it were generated by a different mechanism. Outliers are different from noisy data. Noise is a random error or variance in a measured variable. In general, noise is not interesting in data analysis, including outlier detection. For example, in credit card fraud detection, a customer's purchase behavior can be modeled as a random variable. A customer may generate some "noise transactions" that may seem like "random errors" or "variance," such as by buying a bigger lunch one day or having one more cup of coffee than usual. Such transactions should not be treated as outliers; otherwise, the credit card company would incur heavy costs from verifying that many transactions. The company may also lose customers by bothering them with multiple false alarms. As in many other data analysis and data mining tasks, noise should be removed before outlier detection.

Box-and-whisker plots are a handy way to display data broken into four quartiles, each with an equal number of data values. The box-and-whisker plot doesn't show frequency and it doesn't display each individual statistic, but it clearly shows where the middle of the data lies. It's a nice plot to use when analyzing how your data is skewed.

There are a few important vocabulary terms to know in order to graph a box-and-whisker plot. Here they are:

- **Q1 – quartile 1**, the median of the lower half of the dataset
- **Q2 – quartile 2**, the median of the entire data set
- **Q3 – quartile 3**, the median of the upper half of the dataset
- **IQR – interquartile range**, the difference from Q3 to Q1
- **Extreme Values** – the smallest and largest values in a data set

The "interquartile range", abbreviated "IQR", is just the width of the box in the box-and-whisker plot. That is, IQR = $Q_3 – Q_1$. The IQR can be used as a measure of how spread-out the values are. The IQR is the length of the box in your box-and-whisker plot. An outlier is any value that lies more than one and a half times the length of the box from either end of the box.

Statistics assumes that values are clustered around some central value. The IQR tells how spread out the "middle" values is; it can also be used to tell when some other values are "too far" from the central value. These "too far away" points are called "outliers" because they "lie outside" the range in which we expect them.

Outliers are values that are much bigger or smaller than the rest of the data. These are represented by a dot at either end of the plot. In order to be an outlier, the data value must be:

- Larger than Q3 by at least 1.5 times the interquartile range (IQR), or
- Smaller than Q1 by at least 1.5 times the IQR.

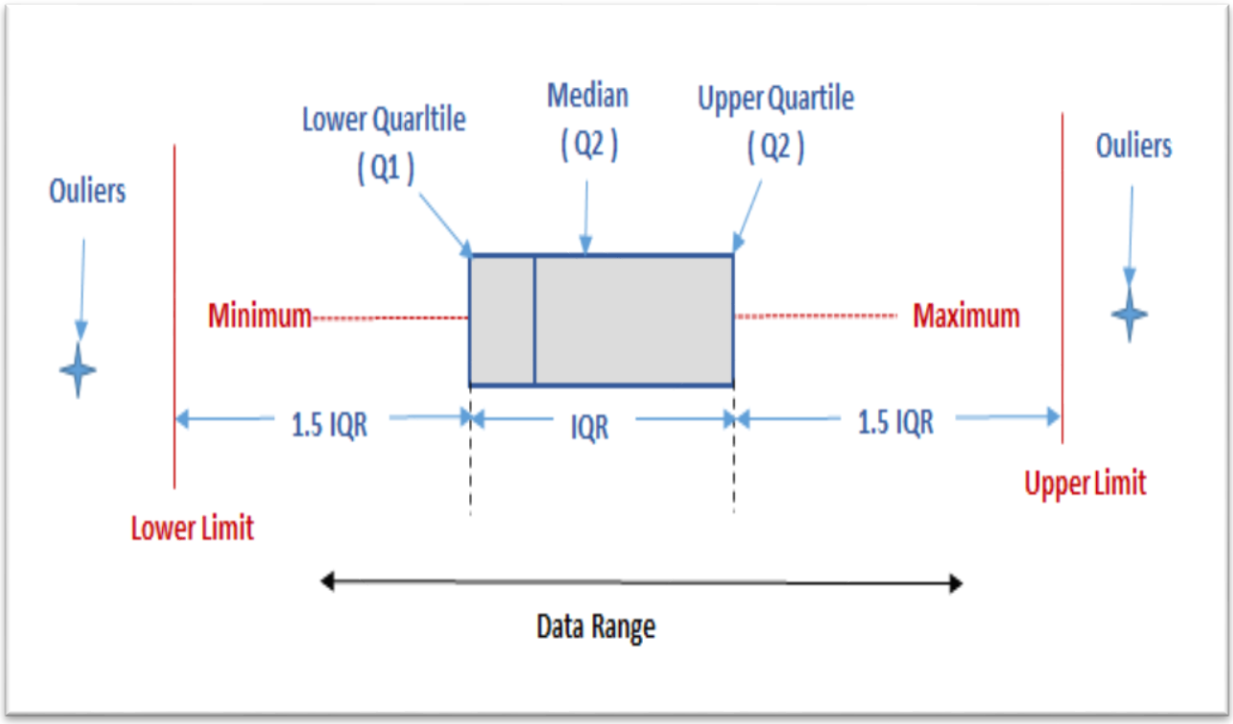


Figure 5.6: Box-and-whisker plot

Outlier Analysis of First Dataset:

We will try to find out the outliers of ‘Log_Sales’ of our first Dataset. The box plot is shown in Figure 5.7.

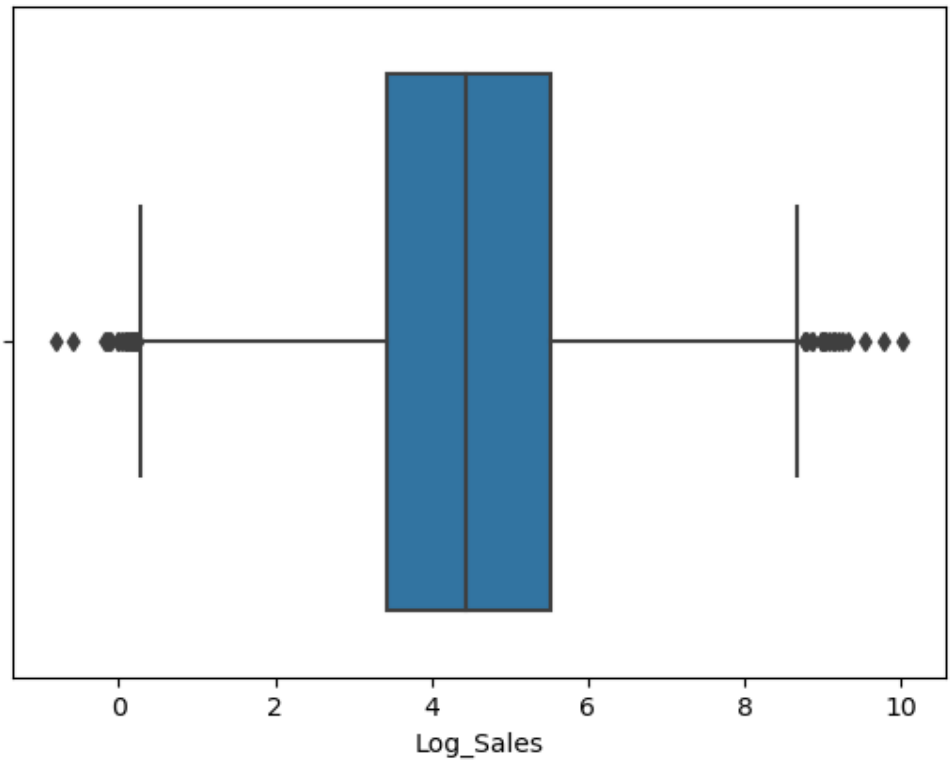


Figure 5.7: Box-and-whisker plot of 'Log_Sales' (First dataset)

From figure 5.7, we see that 'Log_Sales' has some outliers. We will remove this outlier and try to train our model again for predicting the 'ShippingCost' in our first dataset. After removing the outliers, we have got 51238 tuples instead of 51290 tuples. After removing the outliers, the mean RMSE for polynomial regression using degree 1 to 4 are 39.07, 35.39, 30.53 and 28.59 respectively.

Outlier Analysis of Second Dataset:

We check for outliers in 'Log_Sales', 'Log_ShippingCost' and 'Log_Quantity' in our second dataset. The boxplot of 'Log_Sales', 'Log_ShippingCost' and 'Log_Quantity' is shown in Figure 5.8, 5.9 and 5.10 respectively.

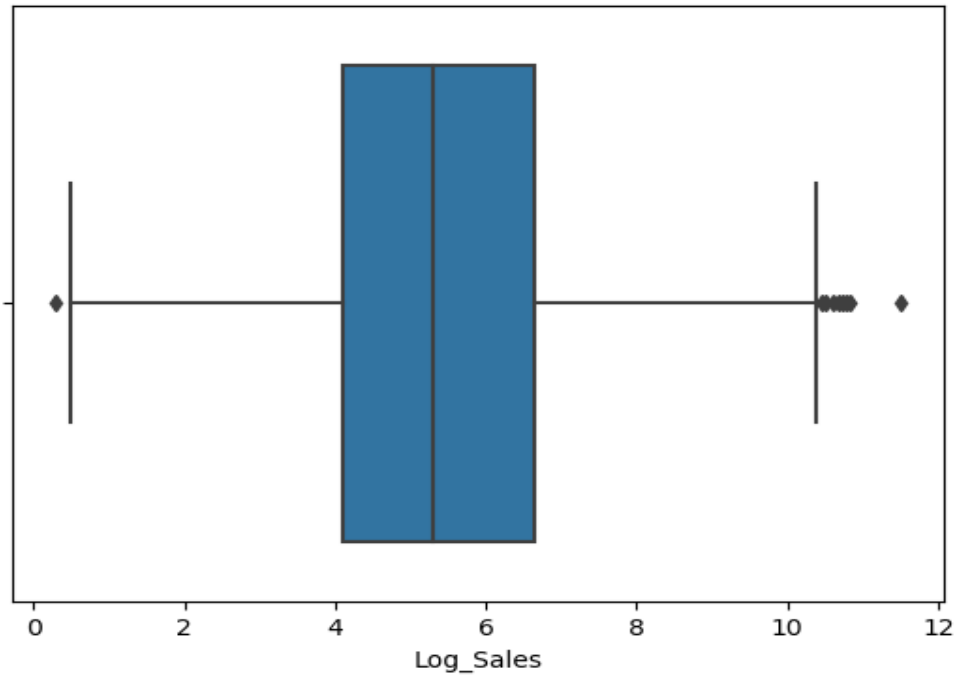


Figure 5.8: Box-and-whisker plot of 'Log_Sales' (Second dataset)

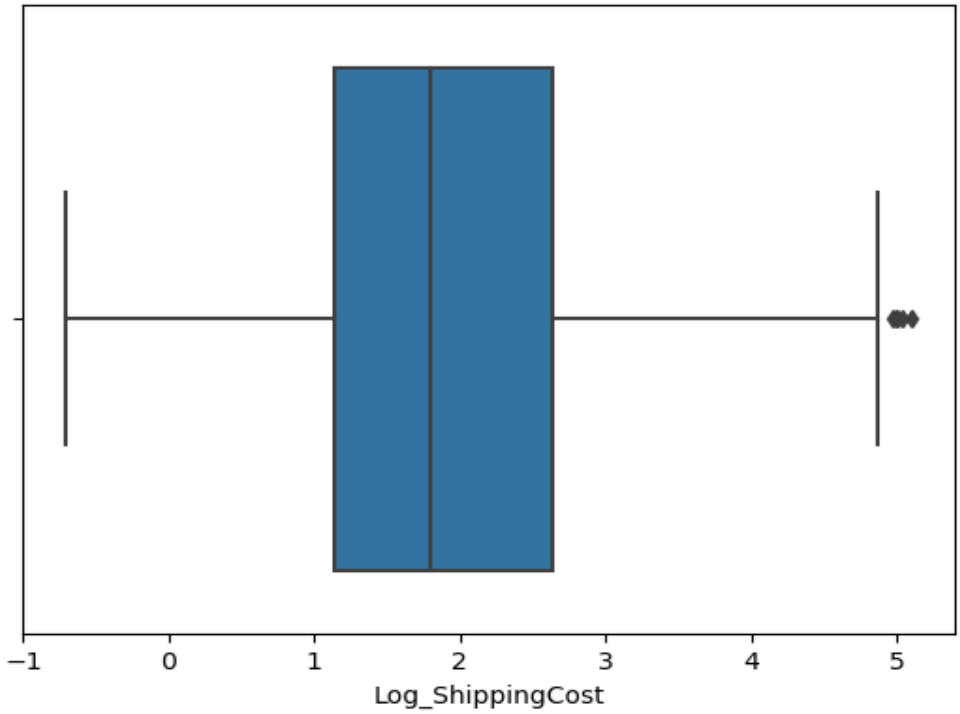


Figure 5.9: Box-and-whisker plot of 'Log_ShippingCost' (Second dataset)

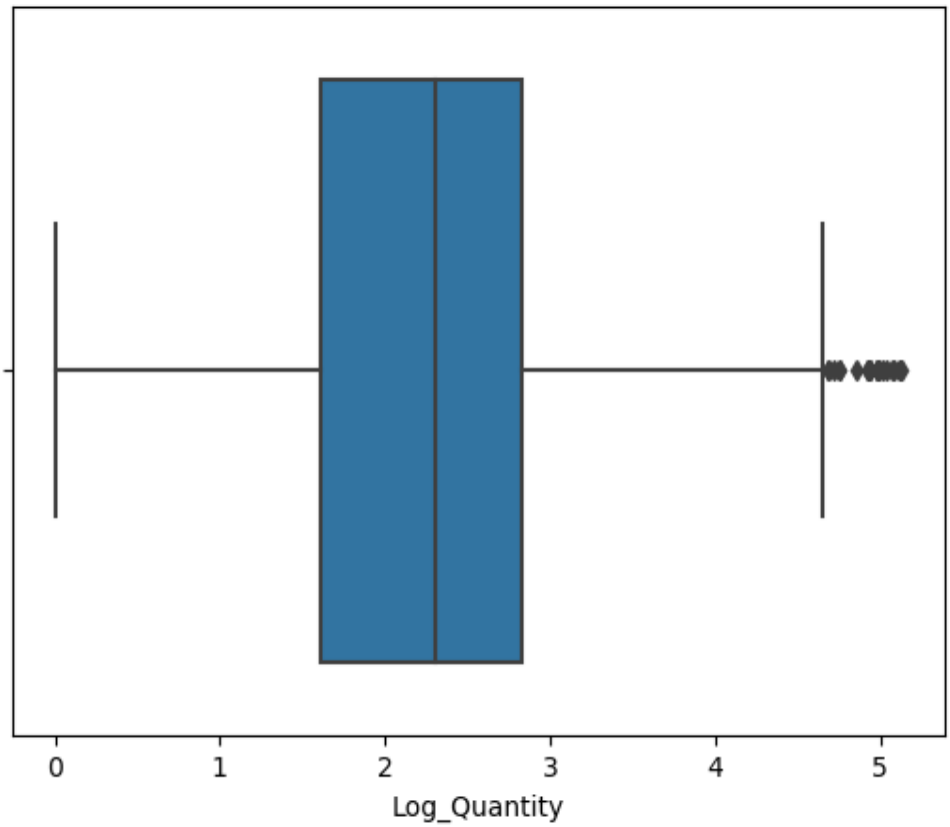


Figure 5.10: Box-and-whisker plot of 'Log_Quantity' (Second dataset)

After removing the outliers of 'Log_Sales', 'Log_ShippingCost' and 'Log_Quantity' from the data, we get 8623 tuples out of 9354 tuples for predicting the 'Profit' and we once again run polynomial regression using degree 1 to 4 and got RMSE value of 871.07, 801.23, 753.13 and 745.33 respectively.

In case of predicting the 'Shipping Cost', we have selected 'Log_sales' and 'Product Base Margin' as our input feature. We have seen that there are some outliers in 'Log_Sales'. After removing these outliers we get 9342 tuples out of 9354 tuples. There are no outliers in 'Product Base Margin'. After removing outliers from the data we once again run polynomial regression using degree 1 to 4 and got the RMSE value of 14.36, 13.83, 13.53 and 13.33 respectively.

5.4 Improving Classification Accuracy

In 5.2.3 and 5.2.4 section, we have implemented logistic regression and K Nearest Neighbors with their default parameters. Now we will tune some parameters and also use our transformed attributes to see how the model changes.

5.4.1 Tuning Inverse of Regularization Strength (C)

Regularization is applying a penalty to increase the magnitude of parameter values in order to reduce overfitting. When you train a model such as a logistic regression model, you are choosing parameters that give you the best fit to the data. This means minimizing the error between what the models predict for your dependent variable given your data compared to what your dependent variable actually is.

The problem comes when you have a lot of parameters (a lot of independent variables) but not too much data. In this case, the model will often tailor the parameter values to idiosyncrasies in your data which means it fits your data almost perfectly. However because those idiosyncrasies don't appear in future data you see, your model predicts poorly.

To solve this, as well as minimizing the error as already discussed, you add to what is minimized and also minimize a function that penalizes large values of the parameters. Most often the function is $\lambda \sum \theta_j^2$, which is some constant λ times the sum of the squared parameter values θ_j^2. The larger λ is the less likely it is that the parameters will be increased in magnitude simply to adjust for small perturbations in the data. In our case however, rather than specifying λ, we will specify C=1/λ. Remember that we use parameter C as our Inverse of Regularization Strength(C). Parameter C = 1/λ.

Lambda (λ) controls the trade-off between allowing the model to increase its complexity as much as it wants with trying to keep it simple. For example, if λ is very low or 0, the model will have enough power to increase its complexity (over fit) by assigning big values to the weights for each parameter. If on the other hand, we increase the value of λ, the model will tend to underfit, as the model will become too simple.

Parameter C will work the other way around. For small values of C, we increase the regularization strength which will create simple models which underfit the data. For big

values of C, we low the power of regularization which implies the model is allowed to increase its complexity and therefore, overfit the data. A valid question to raise is, which C values must we then use? There is a way to check which C values are best. The idea is to understand how each C value affects the accuracy of the training set and the testing set. Remember that our goal is always to create a model that can generalize to unseen data.

Previously we got 62.73% accuracy in predicting the 'OrderPriorityID'. We will use Recursive feature elimination with cross-validation (RFECV) for predicting 'OrderPriorityID'. We have checked the ranking of all attributes using RFECV. Then we have selected the feature which has rank 1. After running RFECV we have got four features ranking one. They are 'DateDif', 'ShipModeID', 'Log_Sales' and 'Log_ShippingCost'. So, we will use these four features as our input feature. Our target variable is 'OrderPriorityID'. We will use 10 fold cross-validation to evaluate the model. Here we will change some parameters of logistic regression. We will set the solver equal to 'sag' which means it will use **Stochastic Average Gradient (SAG)**. It is useful for large datasets. We have set the 'multiclass' parameter of logistic regression in SKLEARN as 'multinomial' and 'max_iter' as 1000. Then we will tune 'C' parameter which is the 'Inverse of Regularization Strength' using the values 0.001, 0.01, 0.1, 1, 10 and 15. After making these changes we have evaluated our model again and get **74.00 %** accuracy at **C=10**. Previously we have seen that 'Log_Sales' and 'Log_ShippingCost' have some outliers. After removing the outliers, we get 51089 tuples out of 51290 tuples. We once again evaluate the model and get **74.14 %** accuracy at **C=15**.

We have also predicted 'ShipModeID' on 5.2.3 section. Now we will also try to evaluate our new model using the same parameters and 'C' values. We will select 'Log_Sales', 'Log_ShippingCost', 'DateDif" as our feature vector and 'ShipModeID' as our target variable. After evaluating the model we get **78.29%** accuracy at **C=0.001**. Then we also remove outliers and evaluate again and get **78.40%** accuracy at **C=0.001**.

5.4.2 Using Different Distance Metrics in KNN

In Section 5.2.4, we have used default distance metric to perform K Nearest Neighbor. Now we will try Manhattan, Chebyshev, Canberra, Bray Curtis and Hamming distance.

The **Manhattan distance** is the simple sum of the horizontal and vertical components whereas the diagonal distance might be computed by applying the Pythagorean Theorem.

In mathematics, **Chebyshev distance** maximum metric is a metric defined on a vector space where the **distance** between two vectors is the greatest of their differences along any coordinate dimension. It is named after Pafnuty **Chebyshev**.

The **Minkowski distance** defines a **distance** between two points in a normed vector space. From the table 5.1, we have seen then there is a parameter '**p**'. Special cases: When p=1, the **distance** is known as the Manhattan **distance**. When p=2, the **distance** is known as the Euclidean **distance.**

The **Canberra metric** is similar to the Manhattan distance (which itself is a special form of the Minkowski distance). The distinction is that the absolute difference between the variables of the two objects is divided by the sum of the absolute variable values prior to summing.

Bray Curtis distance or **Sorensen distance** is a normalization method that is commonly used in botany, ecology field. It views the space as grid similar to the city block distance. The bray Curtis distance has a nice property that if all coordinates are positive its value is between zero and one. Zero bray Curtis represent exact similar coordinate. If both objects are in the zero coordinate, the bray Curtis distance is undefined.

The following Table shows the string metric identifiers and the associated distance metric classes in Scikit-learn:

Identifier	Class name	Distance function
“euclidean”	EuclideanDistance	sqrt(sum((x - y)^2))
“Manhattan”	ManhattanDistance	sum(|x - y|)
“Chebyshev”	ChebyshevDistance	max(|x - y|)
“minkowski”	MinkowskiDistance	sum(|x - y|^p)^(1/p)
“canberra”	CanberraDistance	sum(|x - y| / (|x| + |y|))
“braycurtis”	BrayCurtisDistance	sum(|x - y|) / (sum(|x|) + sum(|y|))
“hamming”	HammingDistance	N_unequal(x, y) / N_tot

Table 5.1: Distance Metrics Formulas

In 5.2.4 section, we have implemented K-nearest neighbor for the prediction of ‘OrderPriorityID’ and ‘ShipModeID’ using Euclidean distance. Now we will try some different distance metrics like Manhattan, Chebyshev, Hamming, Canberra and Bray Curtis distance to evaluate our model. We simply change the ‘metrics’ parameter of K-nearest neighbor classifier in SKLEARN to use different distance metrics. We will select ‘Discount’, ‘CategoryID’, 'DateDif" as our feature vector and ‘OrderPriorityID’ as our target variable. The ‘Discount’ attribute contains some outliers. After removing the outlier we have run our algorithms again and get **61.10%** accuracy using **Manhattan distance**.

For predicting the ‘ShipModeID’, we have selected ‘DateDif’ as our feature vector and ‘ShipModeID’ as our target variable. Here we have also used different distance metrics. The accuracy we have obtained was **81.40%** in all cases.

Chapter 6
Result Analysis and Dataset Visualization

In this section, we will analyse the results we have got from the implementation of chapter five. First, we will see the performance of regression algorithms on our both datasets then we will check the performance and comparison of classification algorithms.

6.1 Result Analysis of Regression Algorithms

In section 5.2.1, we first implemented linear regression for a single particular product only. We get an RMSE value of 5.96993034214e-14 in K-fold Validation. In train-test split we got 6.29793031992e-14. But if we select a different portion of our data again in the train-test split, then we will get different RMSE value. The K-fold validation gives us the best estimation. After fitting the data, we get the coefficient of 'Sales' is 1.00000000e+00,'Quantity' is -3.42900000e+01 and 'Discount' is 1.95399252e-14. The intercept is 1.09690034833e-13. If we put these values in our equation, the equation looks like:

Profit=1.097e-13 + 1.00×Sales - (3.429e+01)×Quantity + (1.95e-15)×Discount

After observing the coefficient of feature attributes, we can say that Discount's coefficient is near to zero. That means this attribute has no impact on the result. If we remove this attribute as our feature and check out our score again then we got the RMSE value of 5.42633456253e-14 which is slightly decreased from the previous model. More precisely, Discount attribute hampers our training. By removing it, we have built a better model.

We have built a model for predicting the profit for all product in the second part of linear regression. We got the RMSE value of 98.76. The reason behind getting higher RMSE is there is no strong relationship between feature attributes and the target variable. In section 5.2.2 we also use polynomial regression using degree 1 to 4 for predicting the profit. After running the polynomial regression using degree 1 to 4 we see that when we are using degree 1 then it gives us a higher RMSE value and it is equal to linear regression RMSE. Polynomial regression with degree 1 is nothing but simple linear regression. As we have already seen that there is no strong relationship here that's why the RMSE is high.

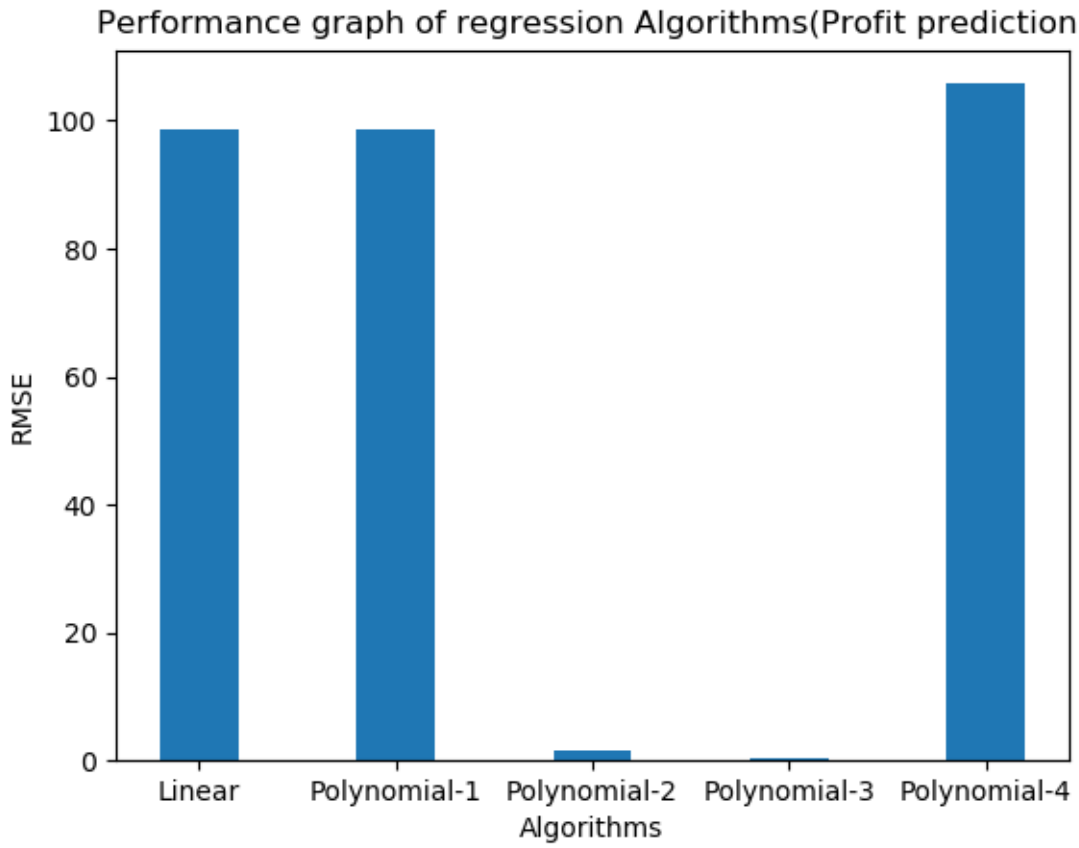


Figure 6.1: Profit prediction for any product (Performance graph)

From the graph, we see that the lowest value we get is 0.34 using polynomial with degree 3. Then the RMSE increases again. We have already known from section 3.2.2 that if we have plenty of observation and there is no linear relationship in our dataset using polynomial regression will give us a better model. It will try to fit a curve that covers the maximum of our observed data. Here we did not have any linear relationship between our target variable and input features. That's why using polynomial regression gives us better model than linear regression. Since the degree is 3 we have a cubic expression.

We have also predicted shipping cost using polynomial regression. We got the RMSE value of 3.31 when we use polynomial with degree 1. We got higher RMSE value 23.18, 261.58 for using polynomial regression with degree 2, 3 respectively. Since we have a little amount of data polynomial regression with a higher degree gives us higher RMSE. One thing should be mentioned that we did not take the features accordingly Pearson correlation. If we take features according to **Pearson Co-efficient** then we get an **RMSE value of 5.7**. In the first case, we took 'RegionID' as our input feature which has no linear relationship with

the target variable but still gave us lower RMSE. But when we take the feature that slightly has a linear relation with the target variable, we get higher RMSE. From here we can conclude that Pearson correlation doesn't always give us best features. We can **combine non-linear** features and can gain better performance.

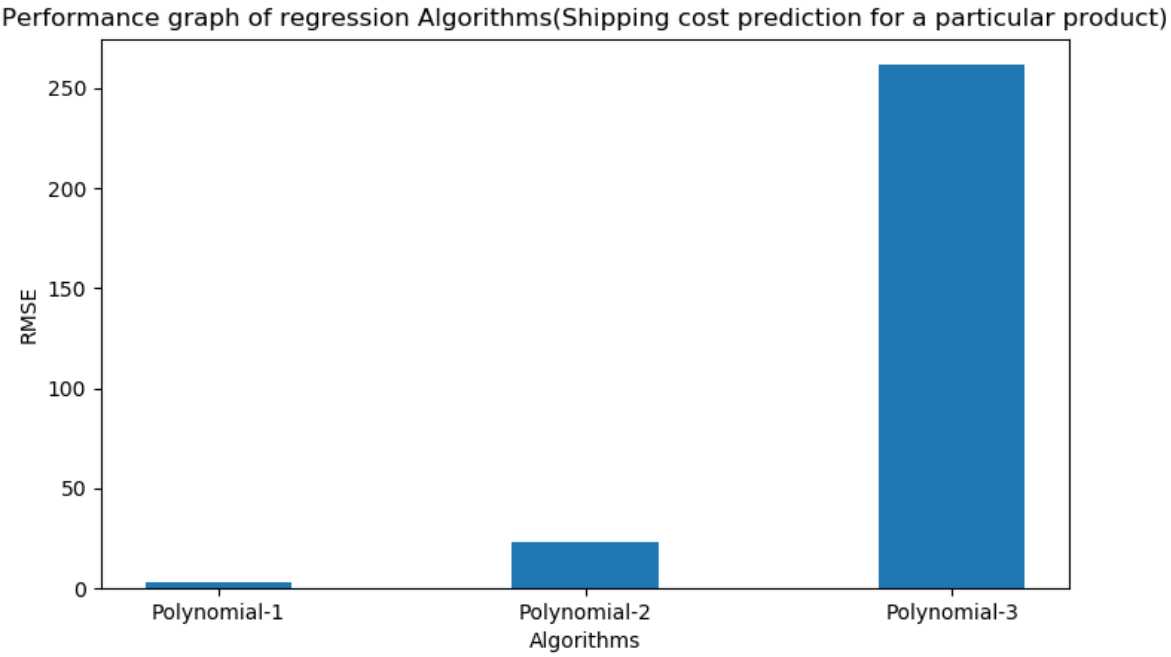


Figure 6.2: Shipping cost prediction for a particular product (Performance graph)

Finally, we have built a model using polynomial regression that can predict the shipping cost of any product. We got reasonable RMSE while predicting shipping cost of a particular product. Now we try to predict the shipping cost of all products. After observing the values of Pearson Correlation, we see that there is no strong relationship with shipping cost. Only 'Sales' have a slightly strong relationship with shipping cost. We took only 'Sales' as our input feature.

After building the model, we see that it gives us lower RMSE value at polynomial regression with degree 2. The lowest RMSE value is 26.27. It seems that the error is high, but this is the best we got. Using other features along with Sales doesn't improve the performance. If we select the features we have used for predicting the shipping cost of a particular product, then the RMSE rises again.

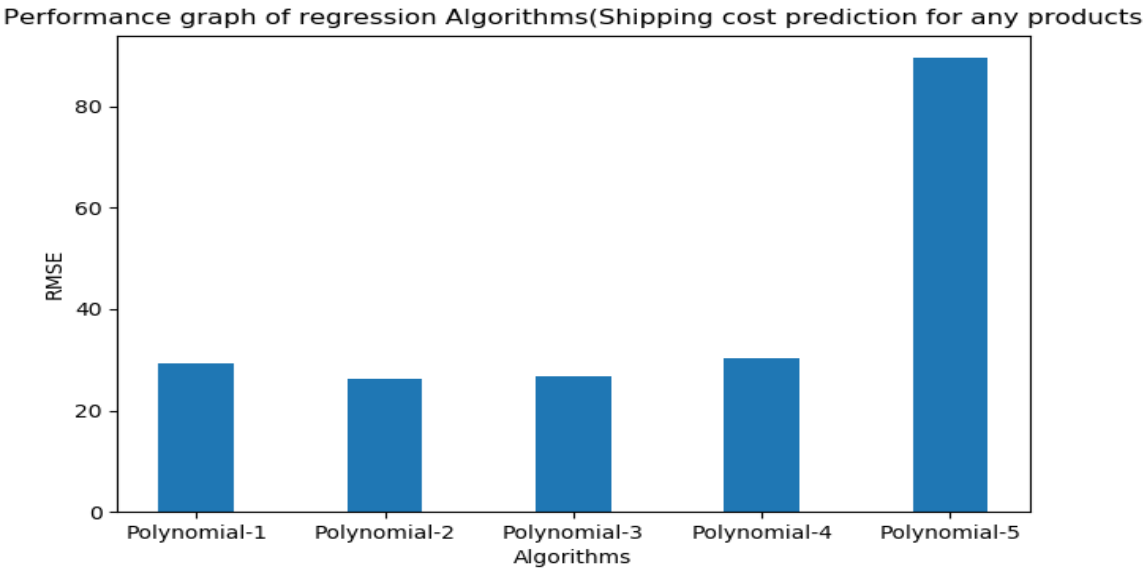


Figure 6.3: Shipping cost prediction for any product (Performance graph)

We have tried to increase the performance using different techniques like variable transformation and outlier analysis and used the same procedure on a new dataset to have a more clear understanding of how these regression algorithms work. The details of our experiment is shown in Table 6.1.

Dataset	Predictions	Techniques	Algorithms performance in RMSE			
			Linear Regression	Polynomial Degree-2	Polynomial Degree-3	Polynomial Degree-4
Dataset-1	Profit prediction (Single product)	Without variable transformation	**6.83e-14**	102.09	1094.23	12042.72
	Shipping cost pred.(Single product)	Without variable transformation	**3.31**	23.17	261.58	2692.18
	Profit prediction (All products)	Without variable transformation	98.76	1.50	**0.34**	105.80
		After variable transformation	116.68	70.88	**1.63**	5.72
		After variable transformation & outlier removal	104.53	68.56	**1.66**	3.068
	Shipping Cost prediction (All products)	Without variable transformation	29.26	**26.27**	26.74	30.31
		After variable transformation	39.46	36.01	30.97	**30.71**
		After variable transformation & outlier removal	39.01	35.39	30.53	**28.59**
Dataset-2	Profit prediction (Single product)	Without variable transformation	**800.214**	8132.20	6665771	8154381
	Shipping cost pred.(Single product)	Without variable transformation	**0.0**	**0.0**	**0.0**	**0.0**
	Profit prediction (All products)	Without variable transformation	**913.08**	924.98	961.37	2313.88
		After variable transformation	931.87	854.33	778.98	**767.42**
		After variable transformation & outlier removal	871.07	801.23	753.13	**745.33**
	Shipping cost prediction (All products)	Without variable transformation	**15.22**	15.41	21.82	48.61
		After variable transformation	14.37	13.84	13.54	**13.34**
		After variable transformation & outlier removal	14.36	13.83	13.53	**13.33**

Table 6.1: Results of regression algorithms on different datasets

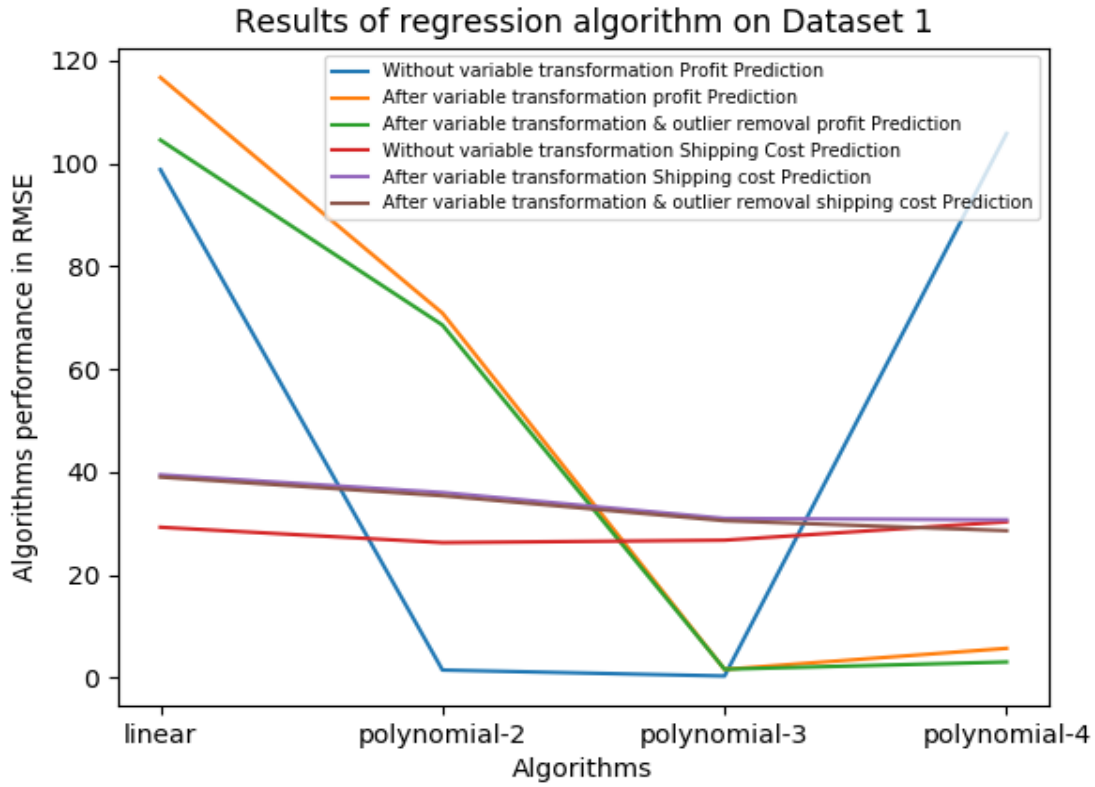


Figure 6.4: Results of regression algorithms on Dataset 1

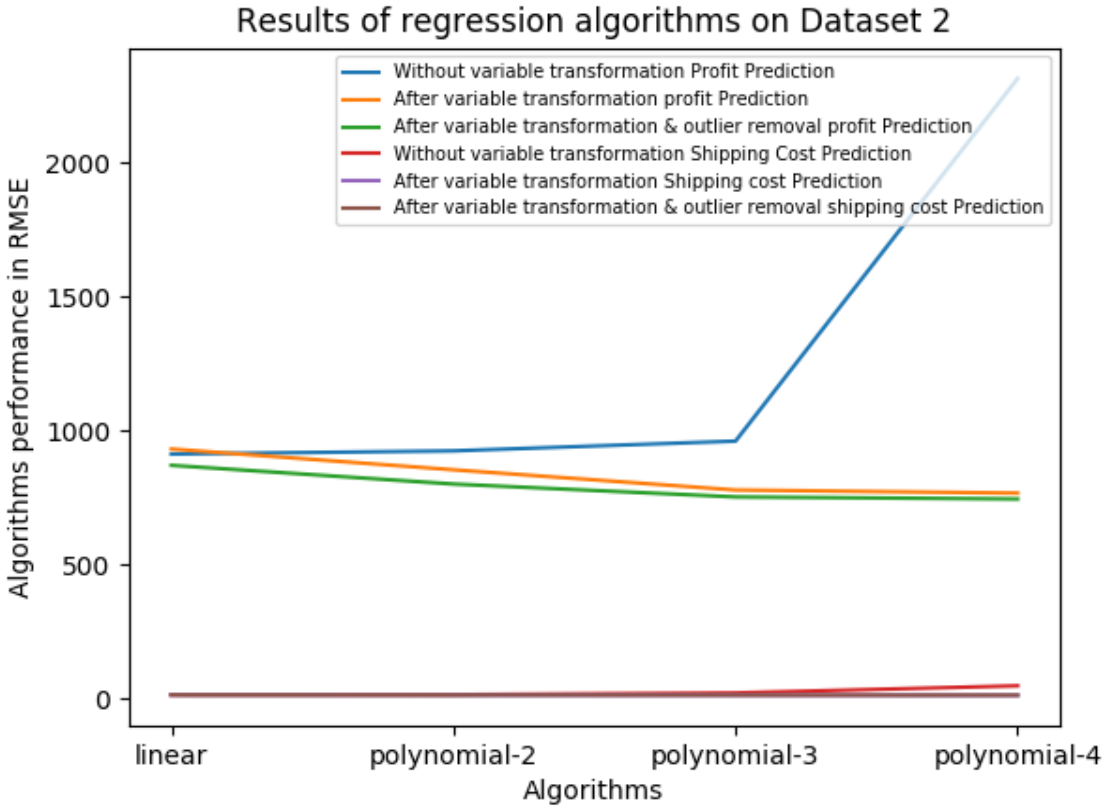


Figure 6.5: Results of regression algorithms on Dataset 2

From the table 6.1 and figure 6.4-5, we observe that in our **first dataset** we could not build a good model for predicting the shipping cost. But after transforming some feature to their natural logarithmic form, we have tried again to predict the 'ShippingCost'. Using the transformed attribute we have got the RMSE to 30.71 using polynomial regression having degree 4. Without variable transformation, we got the RMSE of **26.27**. We have further improved the algorithm by removing the outliers from the dataset. This is caused some loss of data but we get **28.59** RMSE. So in our first dataset, we have seen that performance does not increase. But the performance of higher degree polynomial increases, since logarithmic transformation of a variable reduces the variability of dataset.

We have used **another dataset** to see how these algorithms perform on other data samples. In our second dataset, we have observed that in predicting the profit of a single product, simple linear regression gave relatively higher RMSE. Since we have a small amount of data, polynomial with a higher degree gave us higher RMSE. In our **first dataset**, we have already observed this characteristic. Polynomial regression with a higher degree gives poor performance if we have a small amount of data. Another **caveat** of polynomial regression is that it may cause **overfitting**. It may give good accuracy on training data but when we will test our model using a completely new data, then it can give poor performance.

In our **second dataset**, we have seen that after variable transformation performance of the models increased. In 'Shipping Cost' prediction, RMSE decreased to **13.34**. After removing the outlier the performance remains almost same. There was a very small number of outliers. The dataset contains 9354 tuples. After removing the outliers we got 9342 tuples. Very small amount of data was removed. It does not affect the model that much.

There are some **limitations** of variable transformation especially when we are using natural logarithmic transformation. The natural logarithm of **negative value** is undefined. So if any feature contains negative values then we cannot use logarithmic transformation.

By removing the outlier, we have always got a better model in regression algorithms. Here we also have a limitation. If we have a feature which values vary a lot from some data, Outliers detection will detect many data as outliers that are not actually the outliers. In such cases, we should transform that feature and then try to find out the outliers.

6.2 Result Analysis of Classification Algorithms

In section 5.2.3 & 5.2.4 we have implemented KNN and logistic regression for predicting the order priority and shipping mode of any product from our dataset. We first select important features using RFECV. While predicting the order priority, we can see how the accuracy of logistic regression change from selecting the different number of features from the figure 6.6:

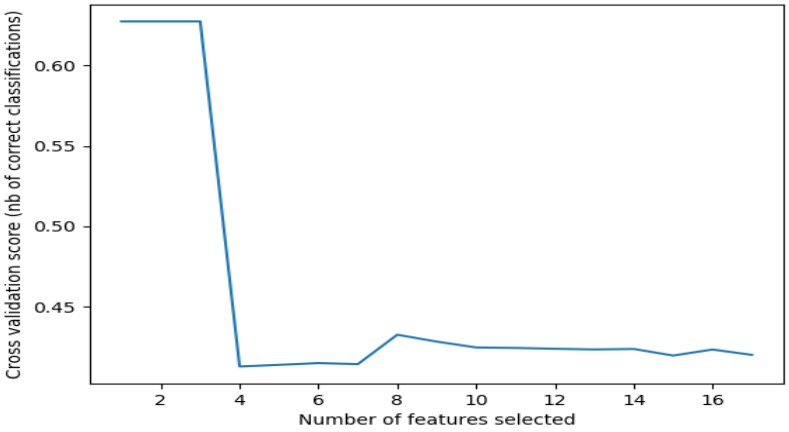


Figure 6.6: Accuracy vs. Number of features selected (Order priority prediction)

While predicting the order priority using KNN, we run the algorithm from 1-700 and select the best k for which we get the maximum accuracy and build the model according to that. The accuracy vs. Value of K ranging from 500 to 600 is shown in figure 6.7.

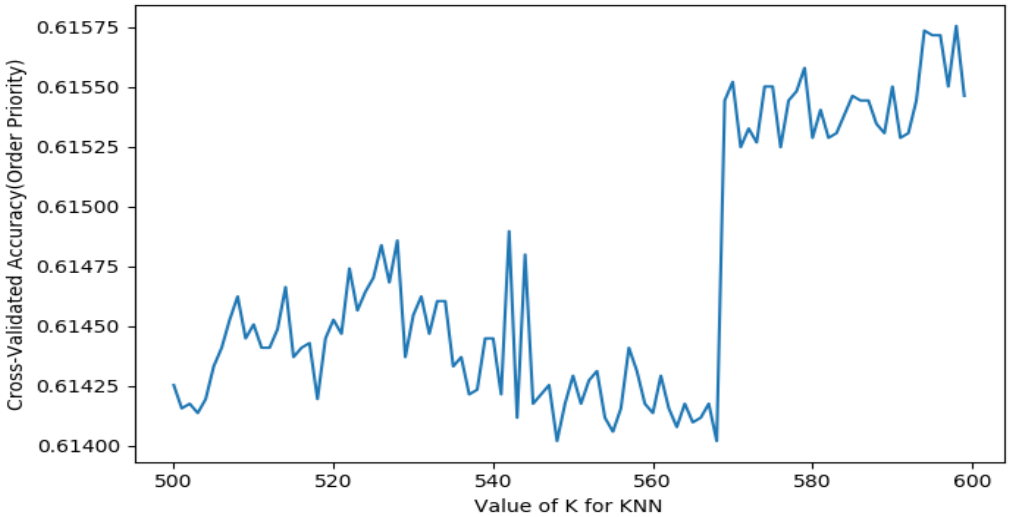


Figure 6.7: Accuracy vs. Value of K for KNN (Order priority prediction)

After running the model for k=1 to 700, we finally get our desired 'K' value 595. The accuracy achieved using **K** is 61.57 %. We took the highest value of K. For KNN models, complexity is determined by the value of K. Lower values of K means more complexity. So, we take the highest value of K that gives us the best accuracy. Logistic regression gives us 62.73 % accuracy.

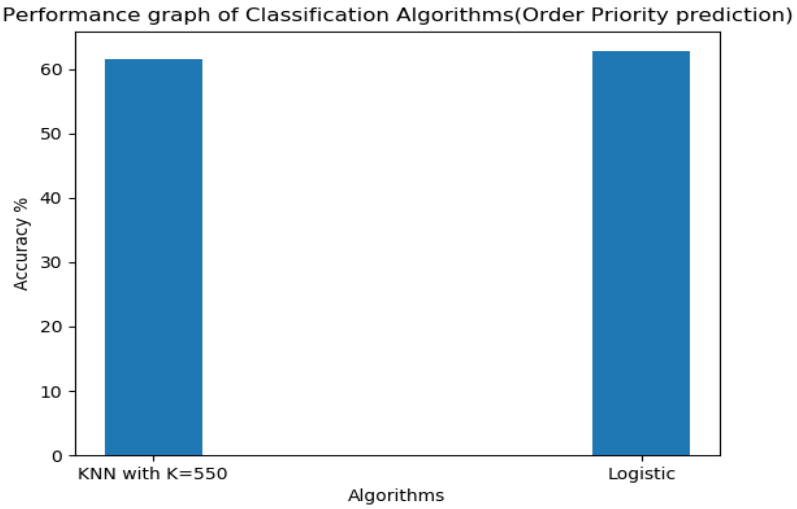


Figure 6.8: Order Priority Prediction

(Performance graph of classification algorithms)

While predicting the shipping mode using KNN, we run the algorithm from 1-60 and select the best k for which we get the maximum accuracy and build the model according to that. The accuracy vs. Value of K ranging from 1 to 60 is shown in figure 6.9:

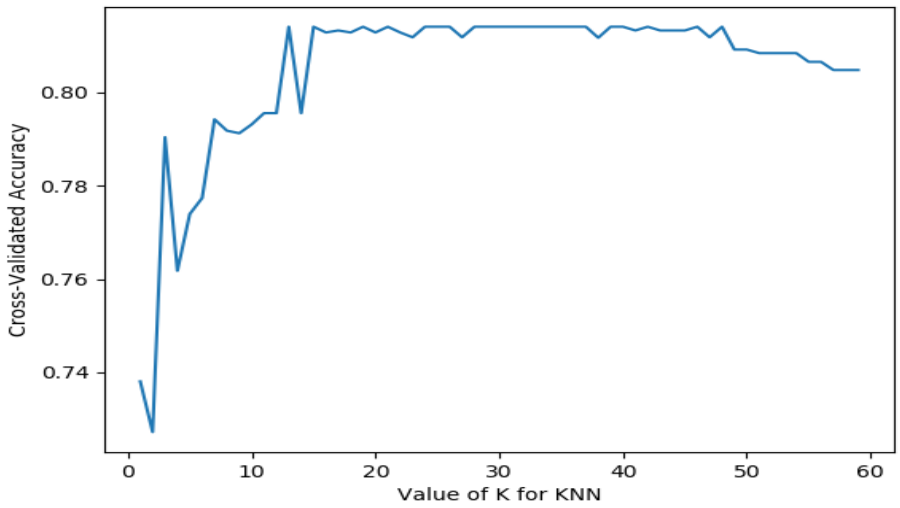


Figure 6.9: Shipping Mode Prediction (Accuracy vs. value of K for KNN)

After running the model for k=1 to 60, we have selected 35 as our k value. The accuracy achieved using k=35 is 81.40 %. Logistic regression gives us 78.12 % accuracy.

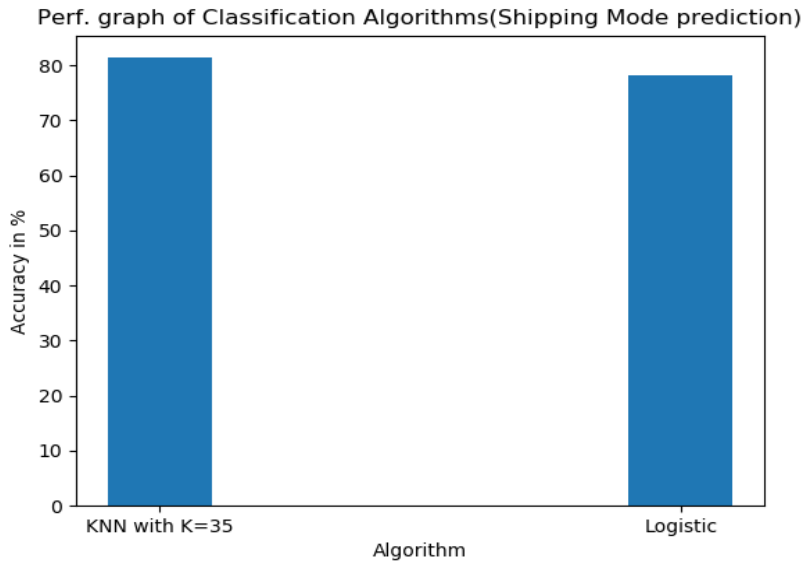


Figure 6.10: Shipping Mode Prediction

(Performance graph of classification algorithms)

K-nearest neighbor and logistic regression both are excellent for classification problem. When we were predicting Order priority we had seen that Logistic regression gave us better accuracy. It also takes little time to build the model. But for KNN it requires more time than logistic regression since it was using 595 neighbors. Using more time, KNN did not give us better accuracy.

In case of predicting the shipment mode, we saw the reverse. KNN gave us better accuracy than logistic regression. The value of K is also reasonable, so it took tolerable time.

Predictions	Inverse of Regularization Strength(C)	Accuracy %	
		After Variable transformation	After Variable transformation & Outlier Removal
Order Priority	0.001	66.81	66.91
	0.01	72.36	72.45
	0.1	73.83	73.97
	1	73.98	74.11
	10	**74.00**	74.13
	15	74.00	**74.14**
Ship Mode	**0.001**	**78.29**	**78.40**
	0.01	78.11	78.18
	0.1	78.09	78.16
	1	78.095	78.15
	10	78.096	78.15
	15	78.096	78.15

Table 6.2: Accuracy of logistic regression after tuning parameters

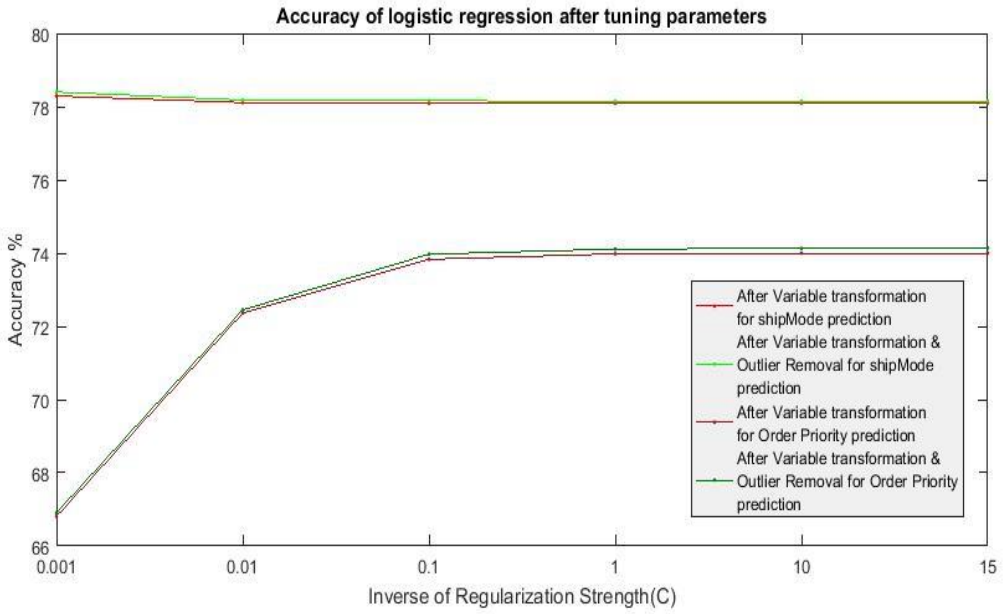


Figure 6.11: Accuracy of logistic regression after tuning parameters

From the above table and graph, we have noticed that the accuracy of predicting the order priority has increased significantly. Previously we have got 62.73% accuracy. After the transformation of some features and tuning some parameters of logistic regression, we have

gained 74% accuracy which is a huge improvement. There were some outliers in our input features. After removing the outliers, we have achieved 74.14 % accuracy. Here while increasing the value of Inverse of Regularization Strength(C), we have also increased the maximum iteration for the solver to converge. The required time to build a model is also increased. But the accuracy has improved significantly so we can overlook the extra execution time.

In case of predicting the 'ShipModeID,' the accuracy is also improved. Here we do not gain huge improvement but using the inverse of regularization strength (C) = 0.001 and removing the outliers we got 78.40 % accuracy. One important thing should be mentioned that if we have used the higher value of the inverse of regularization strength, it will require more time to build the model. The default value of the inverse of regularization strength was 1. Now using C=0.001 we have built a better model which also requires less execution time.

We have also tried different distance metrics to see the effect of K nearest neighbor. The result is shown in table 6.3 and visualized in figure 6.12:

Predictions	Distance Matrices	Accuracy (%)	
		Without Outliers Removing	After Removing Outliers
Order Priority	Manhattan	**61.57**	**61.10**
	Chebyshev	61.55	61.09
	Hamming	60.95	60.51
	Canberra	61.56	61.09
	Bray Curtis	61.55	61.05
Ship Mode	Manhattan	81.40	81.40
	Chebyshev	**81.40**	**81.40**
	Hamming	81.40	81.40
	Canberra	81.40	81.40
	Bray Curtis	81.40	81.40

Table 6.3: Accuracy of K Nearest Neighbors using different distance metrics

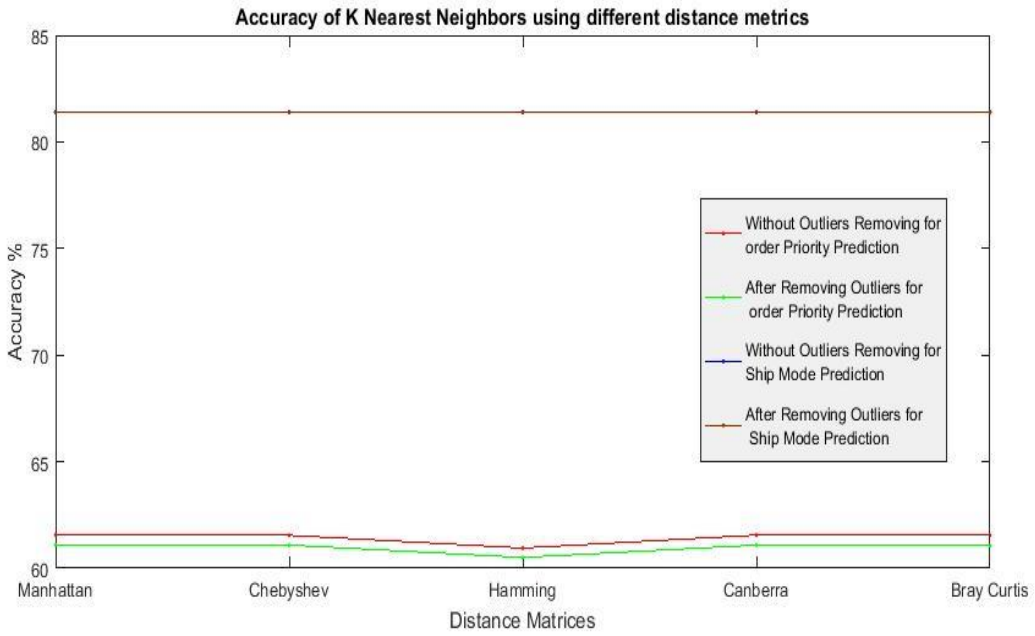


Figure 6.12: Accuracy of K Nearest Neighbors using different distance metrics

Previously we have seen that using the default distance metrics which is Euclidean distance, we had 61.57% accuracy in predicting the 'OrderPriorityID'. After changing the distance metrics we do not have any improvement. But here we see that after removing the outliers of an input feature the accuracy has been decreased in case of K Nearest Neighbors. For the prediction of 'ShipModeID', we have noticed that accuracy does not change after changing the distance metrics because we have used only one input feature. Removing outlier does not affect the accuracy because the input feature does not have any outliers.

The performance of these classification algorithms depends on the nature of data. Some algorithms can discover more insights in data and can improve their accuracy. Some algorithms work best on a certain type of prediction. We are using the same data and features for prediction, but the results are disparate. Same data affects different algorithms different way. Despite of using the same data & features, we get different accuracy.

6.3 Dataset Visualization:

We have used two datasets for implementing the selected algorithms. Now in this section, we will show some core insights of the datasets. Both of them were a transactional dataset. So, sales and profit are the two most important features. Here we will show Yearly sales & profit values and curves as well as monthly sales and profit data.

6.3.1 Dataset-1 Visualization

Year	Month	Profit	Sales	Total Profit	Total Sales
2011	January	8321.80	98898.48	213163.9276	1969236.44
	February	12417.90	91152.15		
	March	15303.56	145729.36		
	April	12902.32	116915.76		
	May	12183.82	146747.83		
	June	23415.24	215207.38		
	July	5585.00	115510.41		
	August	23713.66	207581.49		
	September	35776.88	290214.45		
	October	25963.41	199071.26		
	November	32709.17	298496.53		
	December	40647.98	333925.73		
2012	January	10401.63	135780.72	279639.09876	2388049.52
	February	15000.09	100510.21		
	March	17992.91	163076.77		
	April	17366.96	161052.26		
	May	29876.70	208364.89		
	June	34407.15	256175.69		
	July	15585.38	145236.78		
	August	43573.87	303142.94		
	September	27776.18	289389.16		
	October	30662.88	252939.85		
	November	31820.72	323512.41		
	December	32950.75	338256.96		
2013	January	26810.55	199185.90	368029.5624	3029127.20
	February	23762.49	167239.65		
	March	23433.77	198594.03		
	April	19462.03	177821.31		
	May	28495.69	260498.56		
	June	45478.41	396519.61		

	July	28863.82	229928.95		
	August	31023.66	326488.78		
	September	38905.66	376619.24		
	October	42433.22	293406.64		
	November	48062.99	373989.36		
	December	50202.87	405454.37		
2014	January	28001.38	241268.55	436186.51936	3818708.62
	February	19751.69	184837.35		
	March	37357.26	263100.77		
	April	23782.30	242771.86		
	May	33953.55	288401.04		
	June	43778.60	401814.06		
	July	28035.87	258705.68		
	August	53542.89	456619.94		
	September	67979.45	481157.24		
	October	58209.83	422766.62		
	November	62856.58	555279.02		
	December	46916.52	503143.69		

Table 6.4: Sales and Profit of Dataset 1

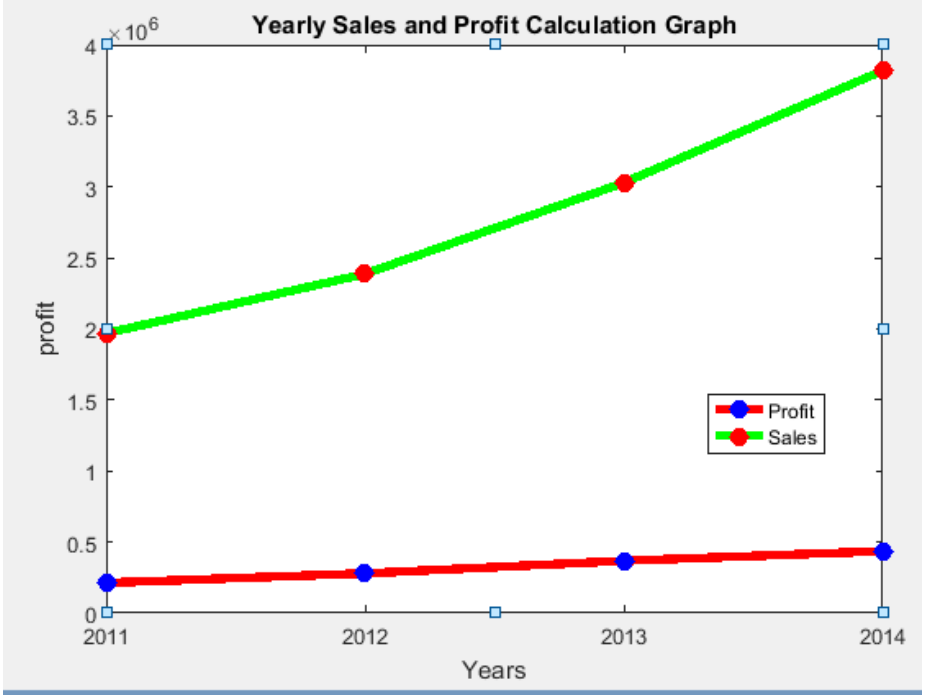


Figure 6.13: Yearly Profit on all transactions (Dataset-1)

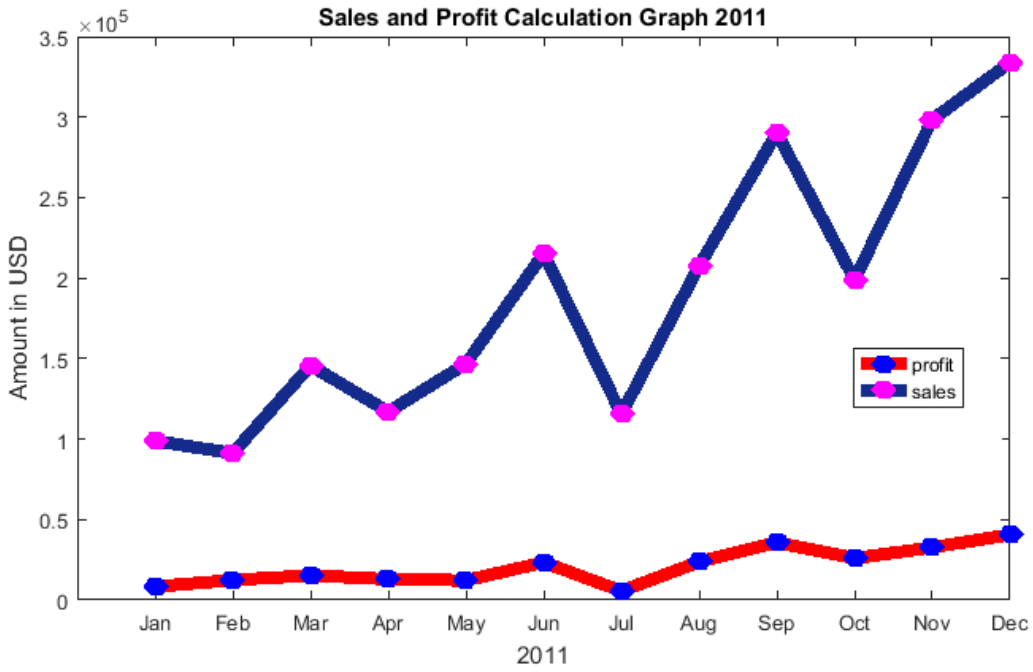


Figure 6.14: Annual Profit for Year 2011 (Dataset-1)

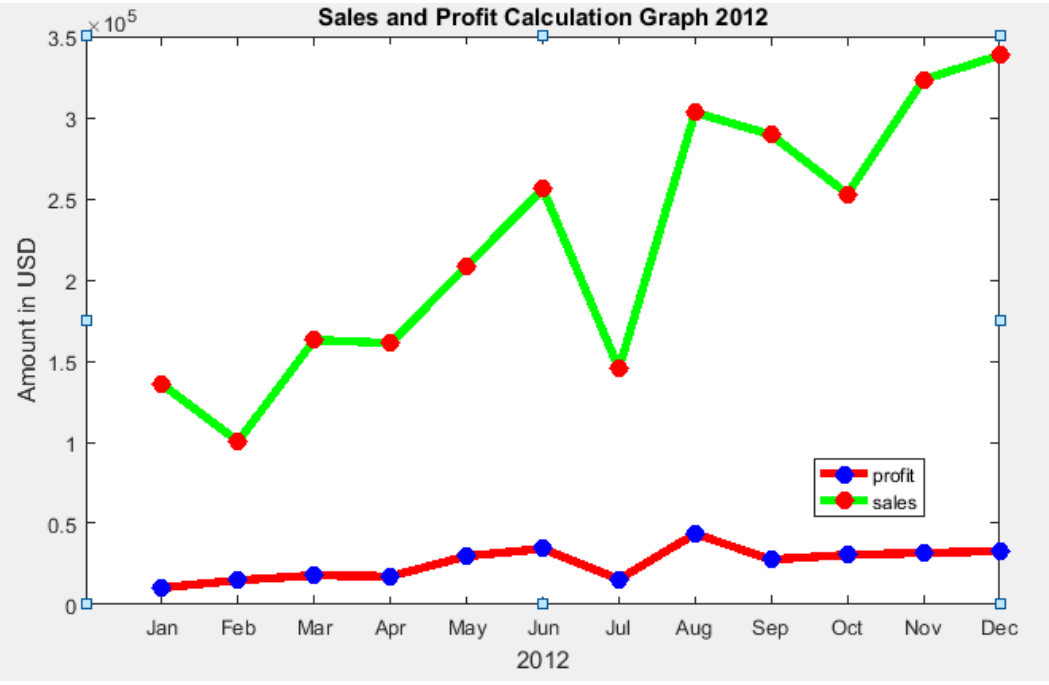


Figure 6.15: Annual Profit for Year 2012 (Dataset-1)

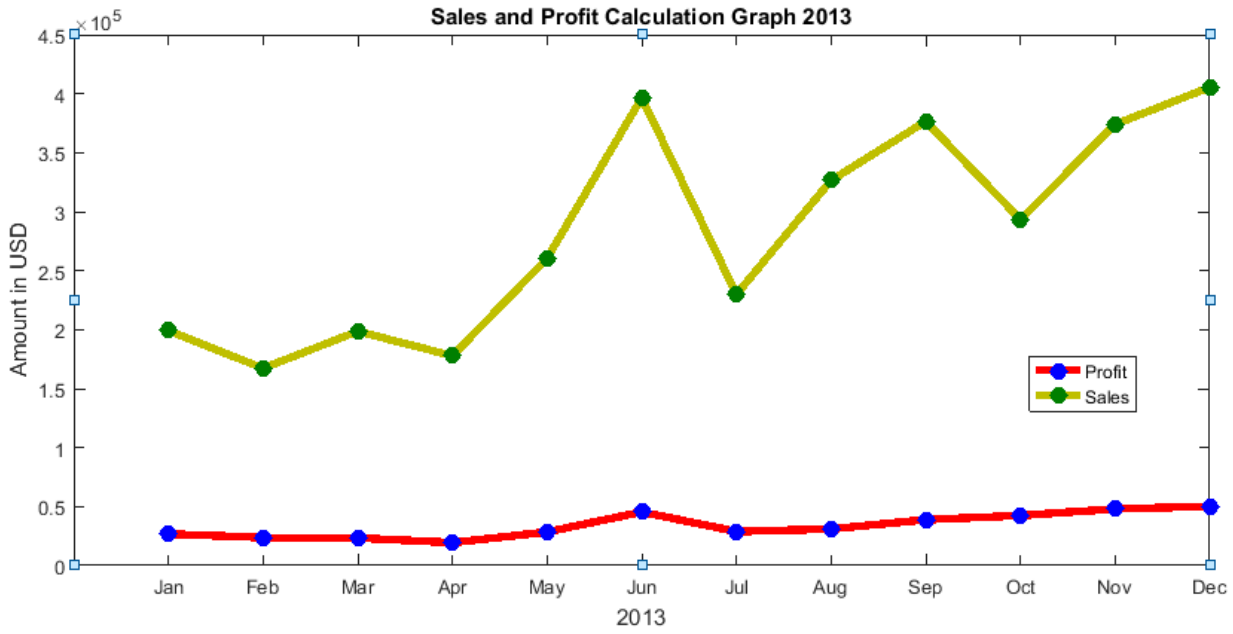


Figure 6.16: Annual Profit for Year 2013 (Dataset-1)

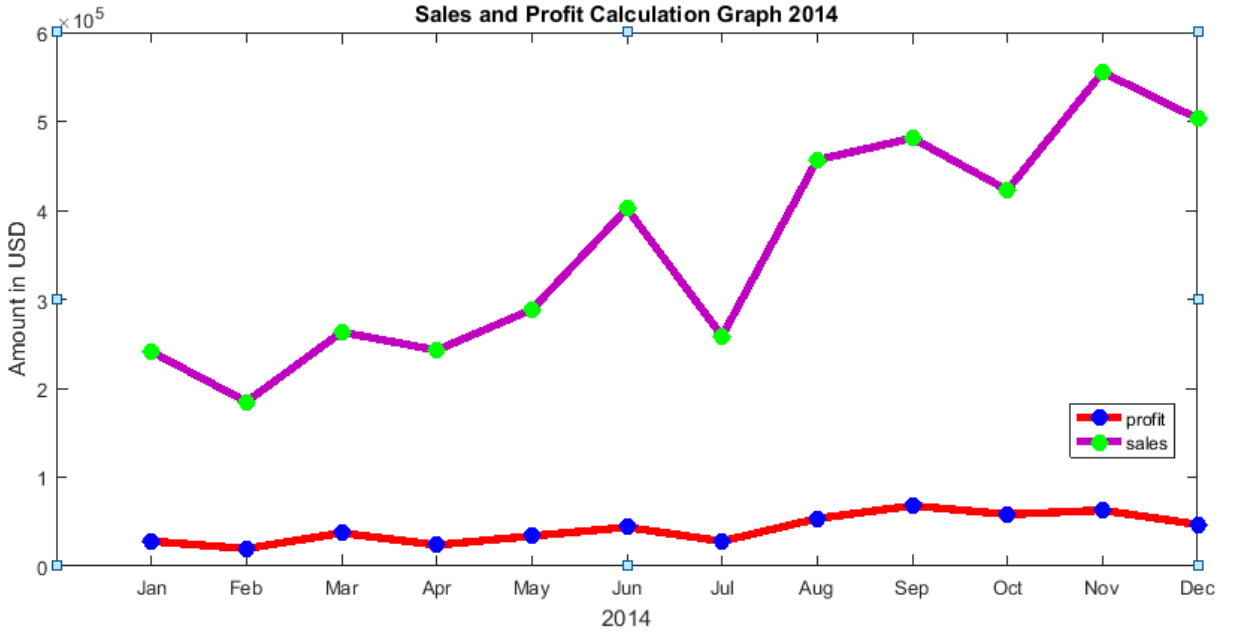


Figure 6.17: Annual Profit for Year 2014 (Dataset-1)

6.3.2 Dataset-2 Visualization

Year	Month	Profit	Sales	Total Profit	Total Sales
2010	January	-2123.25	12047.83	59322.87	303513.17
	February	1767.314	33817.84		
	March	-550.949	26473.45		
	April	5570.327	19785.32		
	May	7858.654	27831.85		
	June	4995.308	18977.32		
	July	973.1872	28266.1		
	August	5778.031	30641.72		
	September	201.0031	23597.69		
	October	12269.16	26462.24		
	November	16418.18	51246.65		
	December	6366.902	27962.85		
2011	January	1247.628	14506.18	39712.10	265541.15
	February	6031.125	20037.37		
	March	2987.198	17751.54		
	April	2217.531	11737.97		
	May	8740.655	66527.95		
	June	1726.895	11964.92		
	July	-1626.11	14539.11		
	August	4821.845	20627.81		
	September	-1016.34	60353.24		
	October	4822.542	48472.64		
	November	7849.015	18972.41		
	December	893.7755	20403.25		
2012	January	213.7319	37088.27	55703.43	376102.16
	February	-4969.5	18131.5		
	March	6015.486	31189.6		
	April	2814.977	30693.06		
	May	10510.12	37593.71		
	June	2859.264	15735.87		
	July	10723.47	51950.51		
	August	-60.2623	21195.61		
	September	5933.172	19924.19		
	October	5675.728	50803.04		
	November	7599.371	32204.22		
	December	14321.04	49516.77		
2013	January	4963.335	27640.64	55403.99	324743.87
	February	4934.765	15973.68		
	March	3929.329	38422.8		
	April	2722.794	38291.53		
	May	1076.991	24514.19		
	June	2312.976	14714.85		
	July	9011.898	36117.9		

	August	17528.71	50780.08		
	September	7247.969	26885.59		
	October	10735.61	35780.36		
	November	-1380.74	27508.17		
	December	-431.671	14999.67		

Table 6.5: Sales and Profit of Dataset 2

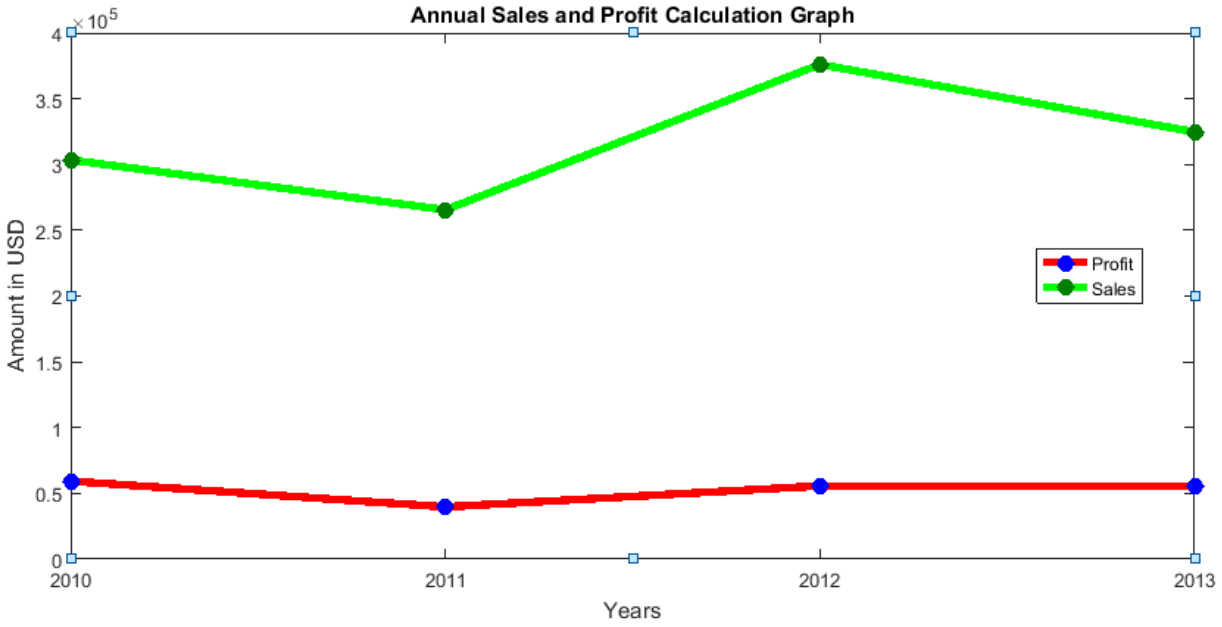


Figure 6.18: Yearly Profit on all transactions (Dataset-2)

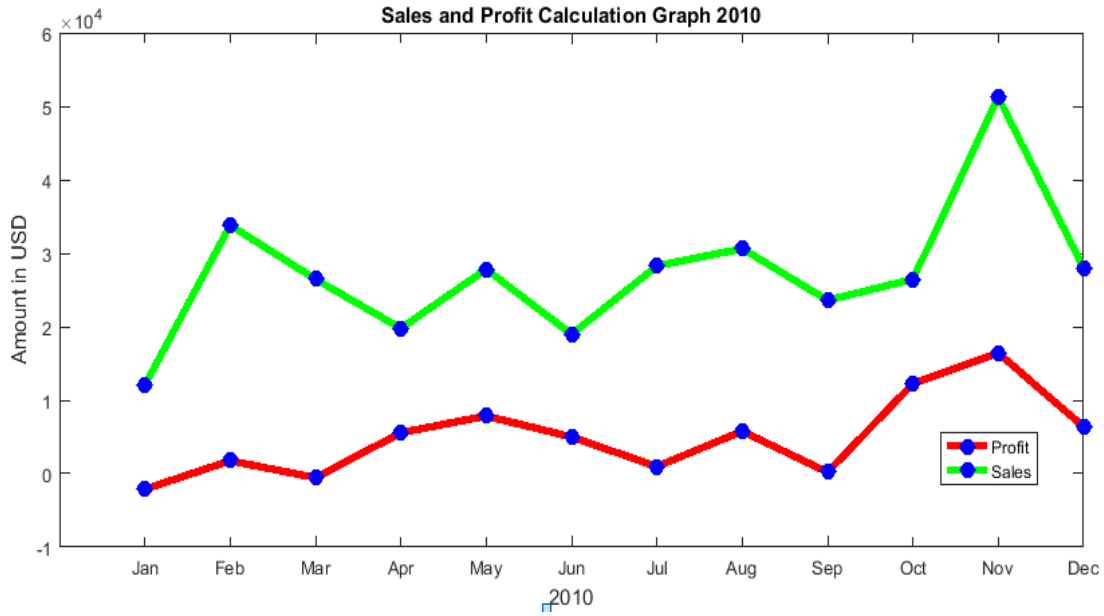


Figure 6.19: Annual Profit for Year 2010 (Dataset-2)

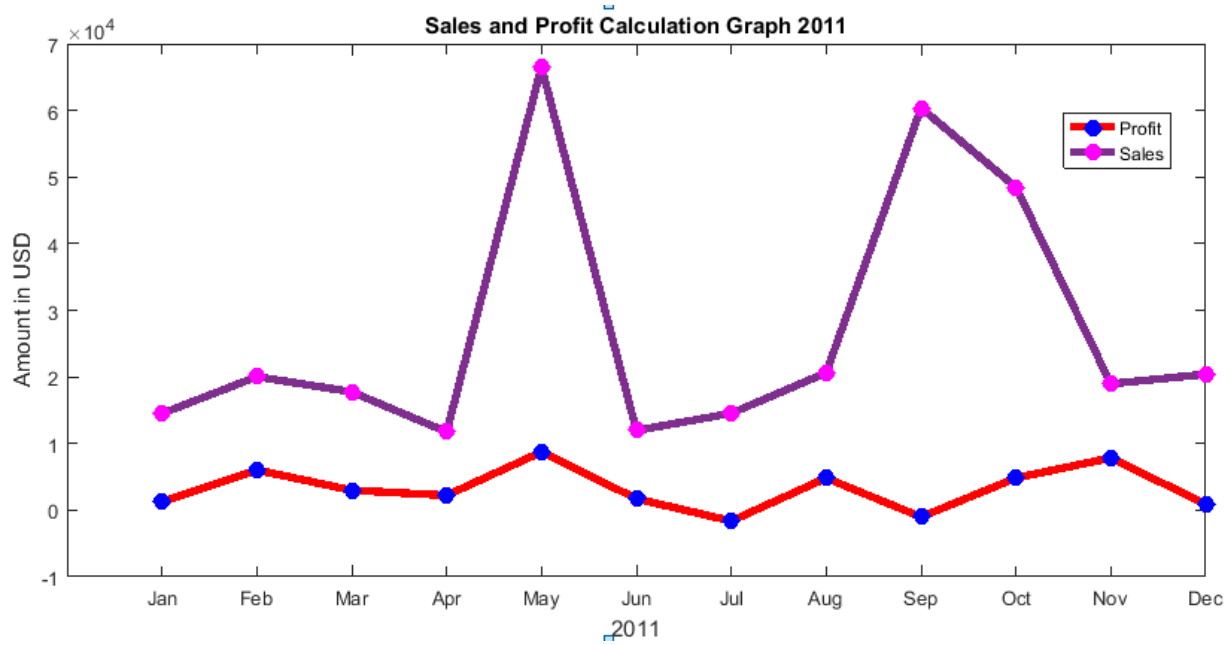


Figure 6.20: Annual Profit for Year 2011 (Dataset-2)

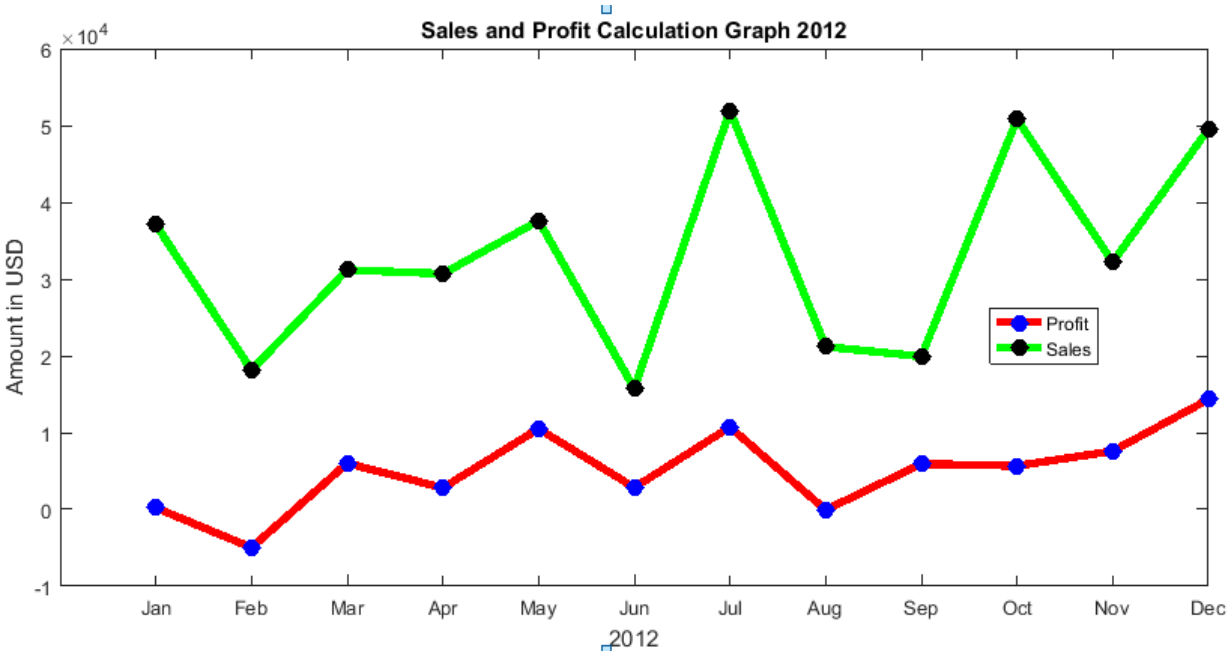


Figure 6.21: Annual Profit for Year 2012 (Dataset-2)

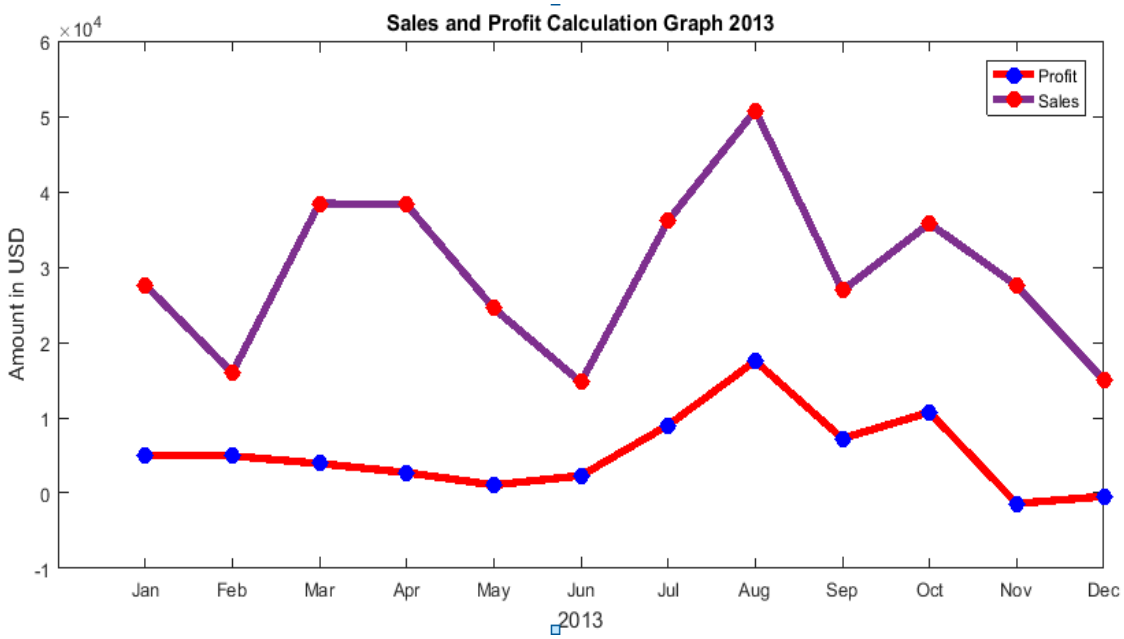


Figure 6.22: Annual Profit for Year 2013 (Dataset-2)

Chapter 7
Conclusion

7.1 General Remarks

Data mining is an emerging field. With the growth of data production and manipulation fields, knowledge of data analysis and mining is becoming one of the most valuable assets day by day. In our work, we tried to have an in-depth understanding of recent data mining techniques and algorithms. To achieve the goals, we have implemented the algorithms and studied their performances. From our study, we have seen that the performance of any specific algorithm mostly depends on the pattern of the data. The same algorithm may provide a different result on different data. So, it is hard to make any generalization to prioritize any specific algorithm over another.

7.2 Specific trends observed for datasets

In both datasets, we have observed that predicting the profit of a single product, simple linear regression gave relatively lower RMSE. Since we have a small amount of data, polynomial regression with a higher degree gave us higher RMSE. Polynomial regression with a higher degree gives poor performance if we have a small amount of data.

In first dataset, there was no significant impact of variable transformation on the performance of the previous result. But the performance of higher degree polynomial increases. Since natural logarithmic transformation removes the variability of dataset, we have gained better performance using variable transformation. In our second dataset, the performance has increased after variable transformation.

There are some derived attributes which were not contained in our initial data set. These derived attribute helps us a lot to build a better model. ‘SellingPriceperunit’, ‘PurchasingPriceperUnit’ and ‘DateDif' help us to build a better regression and classification model.

By removing the outlier, we have always got a better model in regression algorithms. If we have a feature for which values vary a lot from some data, Outliers detection will detect many data as outliers that are not actually the outliers. In these cases, we used logarithm-based transformation of that feature to find out the outliers.

For logistic regression, one important thing should be mentioned that if we have used the higher value of the inverse of regularization strength, it will require more time to build the model. The default value of the inverse of regularization strength was one. By reducing the value of inverse regularization strength, we built a better model which requires less execution time in case of shipping mode prediction.

For K-Nearest Neighbor, after changing the distance metrics, we do not have any significant improvement. But we also saw that after removing the outliers of an input feature the accuracy has been decreased in both datasets. For the predication of 'ShipModeID', we have noticed that accuracy does not change after changing the distance metrics because we have used only one input feature. Removing outlier does not affect the accuracy because the input feature does not have any outliers.

K-nearest neighbor and logistic regression both are excellent for classification problem. But can we make a conclusion that any of them is better than another? The answer is no. When we were predicting Order priority we had seen that Logistic regression gave us better accuracy. It also takes little time to build the model. But for KNN it requires more time than logistic regression since it was using 595 neighbors and it has to save all the data points to build the model. Though it consumed more execution time, KNN did not give us better accuracy.

7.3 Future Scope of Work

We have studied and implemented two popular regression and classification algorithms and analyzed their performances. We have also shown different optimization techniques for the selected algorithms. Our work can be optimized more by applying other distance matrices or using any new optimizing techniques. This can also be improved by using any other superior model training and feature selection strategies. Intelligent heuristic techniques for the specific dataset can also make a huge impact on overall performance.

Moreover, we have used a parametric equation based nearest neighbor algorithm which is computationally costly and performance decreases with the increasing rate of data because this algorithm must save all the data points to perform classification. Support Vector machine-based classifications can overcome such complexity because they perform with the data adjacent to hyper plane and uses Lagrange multiplication techniques.

BIBLIOGRAPHY

1. Jiawei Han and Micheline Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2nd Edition, 2006.

2. Kevin P Murphy. Machine Learning: A Probabilistic Approach. The MIT Press, Cambridge, MA, 2012.

3. S.B. Soumya, N.Deepika. Data Mining With Predictive Analytics for Financial Applications. *International Journal of Scientific Engineering and Applied Science, 2*(1): 310-317, Jan- 2016

4. Usama Fayyad, Gregory Piatetsky-Shapiro, Padhriac Smyth. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In Proc. 1996 AAA *Int. Conf. Knowledge Discovery in Databases (KDD'96)*, pp 82-88, 1996

5. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
   Last Date of visit: 04 December, 2017

6. http://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
   Last Date of visit: 11 November, 2017

7. http://scikit-learn.org/stable/modules/neighbors.html
   Last Date of visit: 15 November, 2017

8. http://scikit-learn.org/stable/auto_examples/linear_model/plot_polynomial_interpolation.html
   Last Date of visit: 12 November, 2017

9. https://en.wikipedia.org/wiki/Cross-validation_(statistics)
   Last Date of visit: 01 October, 2017

10. https://docs.oracle.com/cd/B10501_01/server.920/a96520/concept.html
    Last Date of visit: 01 November, 2017

11. https://docs.oracle.com/cd/B19306_01/datamine.102/b14339/5dmtasks.html
Last Date of visit: 15 April, 2018

12. https://www.tableau.com/sites/default/files/training/global_superstore.zip
Last Date of visit: 04 June, 2017

13. https://docs.oracle.com/cd/B10501_01/server.920/a96520/concept.html
Last Date of visit: 15 April, 2018

14. David A. Freedman. Statistical Models: Theory and Practice. Cambridge Universit y Press. p. 26, 2009

15. Hilary L. Seal.The historical development of the Gauss linear model". Biometrika. 54 (1/2): 1–24, 1967

16. Altman, N.S. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician. 46 (3): 175–185. 1992

17. Ali Radhi Al Essa, Christian Bach. Data Mining and Warehousing. ASEE 2014 Zone I Conference. University of Bridgeport, Bridgeport, CT, USA, Apr-2014

18. https://www.kaggle.com/joparga3/2-tuning-parameters-for-logistic-regression
Last Date of visit: 15 April, 2018

19. https://community.tableau.com/servlet/JiveServlet/download/438262-79453/Sample-Superstore-Subset-Excel.xlsx
Last Date of visit: 30 March, 2018

20. http://scikitlearn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html
Last Date of visit: 17 April, 2018

21. http://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
Last Date of visit: 17 April, 2018

APPENDIX

Important code segments of our work are shown below.

Code Snippet for Building Dimension Table:

```
# Dimesion Table For Product
ProductData=pd.read_excel('DB.xls',usecols=['Product ID','Product Name','Sub-
Category','Sales','Quantity','Discount','Profit'])
ProductData.rename(columns={'Product ID':'ProductID','Product
Name':'ProductName','Sub-Category':'SubCategory'},inplace=True)
ProductData=ProductData.drop_duplicates(subset=['ProductID','ProductName'])

ProductData.ProductName=ProductData.ProductName.str.replace('"','')
ProductData.ProductName=ProductData.ProductName.str.replace('/',' ')
ProductData.ProductName=ProductData.ProductName.str.replace('”','')

UnitPrice=ProductData.Sales/ProductData.Quantity
SellPrice=UnitPrice/(1.0-ProductData.Discount)
ProductData['SellingPricePerUnit']=SellPrice

PurchasingPrice=(ProductData.Sales-ProductData.Profit)/ProductData.Quantity
ProductData['PurchasingPricePerUnit']=PurchasingPrice
#ProductData.set_index(['ProductID','ProductName'],inplace=True)
ProductData=ProductData.drop(['Sales','Quantity','Discount','Profit'],axis=1)

SubCat=pd.read_csv('dimSubCategory.csv')
MergeData = pd.merge(ProductData,SubCat, how='left',on='SubCategory')

MergeData=MergeData.drop(['SubCategory','CategoryID'],axis=1)
MergeData.set_index(['ProductID','ProductName'],inplace=True)
MergeData.to_csv('dimProduct.csv')

#Dimesion Table For Product
ProdData=pd.read_csv('dimProduct.csv',encoding='latin-1')
ProdData.index=range(0,len(ProdData))
ProdData['ProdID']=ProdData.index
NewID=ProdData.ProdID.astype(str)
NewID='P'+NewID
ProdData['ProdID']=NewID
ProdData.set_index('ProdID',inplace=True)
print(ProdData)
ProdData.to_csv('dimNewProduct.csv')
```

Code Snippet for Building the Fact Table:

```
import pandas as pd

FactTable=pd.read_excel('DB.xls',encoding='latin-1')
FactTable.rename(columns={'Order ID':'OrderID','Order Date':'OrderDate','Ship
Date':'ShipDate','Ship Mode':'ShipMode','Customer ID': 'CustomerID','Product
ID':'ProductID','Sub-Category':'SubCategory','Shipping Cost':'ShippingCost',
'Order Priority': 'OrderPriority','Customer Name': 'CustomerName','Row
ID':'RowID','Product Name':'ProductName'},inplace=True)
```

```
OrderPrio=pd.read_csv('dimOrderPriority.csv')
MergeData = pd.merge(FactTable, OrderPrio, how='left', on=['OrderPriority'])
MergeData=MergeData.drop('OrderPriority',axis=1)

ShipMode=pd.read_csv('dimShipMode.csv')
MergeData = pd.merge(MergeData, ShipMode, how='left', on=['ShipMode'])
MergeData=MergeData.drop(['ShipMode','Postal Code'],axis=1)


GetOrderDate=pd.read_csv('dimOrderDate.csv')
MergeData.OrderDate=pd.to_datetime(MergeData.OrderDate)
GetOrderDate.OrderDate=pd.to_datetime(GetOrderDate.OrderDate)
MergeData.ShipDate=pd.to_datetime(MergeData.ShipDate)
MergeData = pd.merge(MergeData, GetOrderDate, how='left', on=['OrderDate'])
MergeData=MergeData.drop(['OrderDate','Year','Month','Quarter'],axis=1)

GetShipDate=pd.read_csv('dimShipDate.csv')
GetShipDate.ShipDate=pd.to_datetime(GetShipDate.ShipDate)
MergeData.ShipDate=pd.to_datetime(MergeData.ShipDate)
MergeData = pd.merge(MergeData,GetShipDate, how='left', on=['ShipDate'])
MergeData=MergeData.drop(['ShipDate','Year','Month','Quarter'],axis=1)
print(MergeData.shape)

GetCity=pd.read_csv('dimCity.csv',encoding='latin-1')
GetState=pd.read_csv('dimState.csv',encoding='latin-1')
merg=pd.merge(GetCity,GetState,how='left',left_on='StateID',right_on='StateID')
GetContry=pd.read_csv('dimCountry.csv')
merg=pd.merge(merg,GetContry,how='left',left_on='CountryID',right_on='CountryID')
MergeData = pd.merge(MergeData, merg,
how='left',left_on=['City','State','Country'],right_on=['City','State','Country'])
print(MergeData.shape)
MergeData=MergeData.drop(['City','StateID','CustomerName','Segment','State','Countr
y','StateID','CountryID'],axis=1)


GetRegion=pd.read_csv('dimRegion.csv')
GetMarket=pd.read_csv('dimMarket.csv')
Merg=pd.merge(GetRegion,GetMarket,how='left',left_on='MarketID',right_on='MarketID'
)
MergeData = pd.merge(MergeData,Merg,
how='left',left_on=['Region','Market'],right_on=['Region','Market'])
MergeData=MergeData.drop(['Region','Market','MarketID'],axis=1)
print(MergeData.shape)

MergeData=MergeData.drop(['Category','SubCategory'],axis=1)
GetProduct=pd.read_csv('dimNewProduct.csv',encoding='latin-1')
MergeData.ProductName=MergeData.ProductName.str.replace('"','')
MergeData.ProductName=MergeData.ProductName.str.replace('/',' ')
MergeData.ProductName=MergeData.ProductName.str.replace('”','')
MergeData=pd.merge(MergeData,GetProduct,how='left',on=['ProductID','ProductName'])


MergeData=MergeData.drop(['ProductID','ProductName','SellingPricePerUnit','Purchasi
ngPricePerUnit'],axis=1)
MergeData=MergeData[['RowID','OrderID','OrderDateID','ShipDateID','ShipModeID','Cus
tomerID','ProdID','CityID',
'RegionID','Sales','Quantity','Discount','Profit','ShippingCost','OrderPriorityID']
]
```

```
MergeData.set_index('RowID',inplace=True)
print(MergeData.columns)
print(MergeData.shape)

MergeData.to_csv('FactTab.csv')
```

Code Snippet for Building the Numeric Dataset:

```
import pandas as pd
import numpy as np

Data=pd.read_csv('FactTab.csv',usecols=['RowID','OrderID','ProdID','OrderDateID','S
hipDateID','Sales','Quantity','Discount','Profit','ShippingCost','CityID','RegionID
',
                                        'OrderPriorityID','ShipModeID'])

# df=Data.groupby(['ProdID']).ProdID.agg(['count'])
# print(df['count'].sort_values())
getProduct=pd.read_csv("dimNewProduct.csv",encoding='latin-1')
Data=pd.merge(Data,getProduct,how='left',left_on='ProdID',right_on='ProdID')

Data.ProdID=Data.ProdID.str.replace('P','0')
Data.ProdID=pd.to_numeric(Data.ProdID,errors='coerce').fillna(0).astype(np.int64)

Data.OrderPriorityID=Data.OrderPriorityID.str.replace('OP','0')
Data.OrderPriorityID=pd.to_numeric(Data.OrderPriorityID,errors='coerce').fillna(0).
astype(np.int64)

Data.ShipModeID=Data.ShipModeID.str.replace('SM','0')
Data.ShipModeID=pd.to_numeric(Data.ShipModeID,errors='coerce').fillna(0).astype(np.
int64)

getCity=pd.read_csv('dimCity.csv',encoding='latin-1')
getState=pd.read_csv('dimState.csv',encoding='latin-1')
getCountry=pd.read_csv('dimCountry.csv')
MergeData=pd.merge(getCity,getState,how='left',left_on='StateID',right_on='StateID'
)
MergeData=pd.merge(MergeData,getCountry,how='left',left_on='CountryID',right_on='Co
untryID')

Data=pd.merge(Data,MergeData,how='left',left_on='CityID',right_on='CityID')

Data=Data.drop(['City','State','Country'],axis=1)
getRegion=pd.read_csv('dimRegion.csv')
getMarket=pd.read_csv('dimMarket.csv')

getRegion=pd.merge(getRegion,getMarket,how='left',left_on='MarketID',right_on='Mark
etID')

getRegion=getRegion.drop(['Region','Market'],axis=1)
Data=pd.merge(Data,getRegion,how='left',left_on='RegionID',right_on='RegionID')

Data.MarketID=Data.MarketID.str.replace('MA','0')
Data.MarketID=pd.to_numeric(Data.MarketID,errors='coerce').fillna(0).astype(np.int6
4)

Data.CityID=Data.CityID.str.replace('CI','0')
Data.CityID=pd.to_numeric(Data.CityID,errors='coerce').fillna(0).astype(np.int64)

Data.RegionID=Data.RegionID.str.replace('RE','0')
Data.RegionID=pd.to_numeric(Data.RegionID,errors='coerce').fillna(0).astype(np.int6
4)

Data.StateID=Data.StateID.str.replace('ST','0')
Data.StateID=pd.to_numeric(Data.StateID,errors='coerce').fillna(0).astype(np.int64)
```

```
getShipdate=pd.read_csv('dimShipDate.csv',usecols=['ShipDateID','ShipDate'])
getOrderdate=pd.read_csv('dimOrderDate.csv',usecols=['OrderDateID','OrderDate'])
getShipdate.ShipDate=pd.to_datetime(getShipdate.ShipDate)
getOrderdate.OrderDate=pd.to_datetime(getOrderdate.OrderDate)

Data=pd.merge(Data,getShipdate,how='left',left_on='ShipDateID',right_on='ShipDateID
')
Data=pd.merge(Data,getOrderdate,how='left',left_on='OrderDateID',right_on='OrderDat
eID')

diff=Data.ShipDate-Data.OrderDate

Data['DateDif']=diff
Data.DateDif=Data.DateDif.astype(str)
Data.CountryID=Data.CountryID.str.replace('CO','0')
Data.CountryID=pd.to_numeric(Data.CountryID,errors='coerce').fillna(0).astype(np.in
t64)

Data.DateDif=Data.DateDif.str.replace(' days 00:00:00.000000000','')
Data.DateDif=pd.to_numeric(Data.DateDif,errors='coerce').fillna(0).astype(np.int64)

getSubCat=pd.read_csv('dimSubCategory.csv')
getCat=pd.read_csv('dimCategory.csv')
Data=pd.merge(Data,getSubCat,how='left',left_on='SubCategoryID',right_on='SubCatego
ryID')
Data=pd.merge(Data,getCat,how='left',left_on='CategoryID',right_on='CategoryID')

Data.CategoryID=Data.CategoryID.str.replace('CA','0')
Data.CategoryID=pd.to_numeric(Data.CategoryID,errors='coerce').fillna(0).astype(np.
int64)

Data.SubCategoryID=Data.SubCategoryID.str.replace('SC','0')
Data.SubCategoryID=pd.to_numeric(Data.SubCategoryID,errors='coerce').fillna(0).asty
pe(np.int64)
Data=Data.drop(['ShipDate','OrderDate','SubCategory','Category','ProductName','Prod
uctID'],axis=1)

Data['Log_Sales']=np.log(Data.Sales);
Data['Log_ShiipingCost']=np.log(Data.ShippingCost);
Data['Log_Profit']=np.log(Data.Profit);
Data['Log_Quantity']=np.log(Data.Quantity);

print(Data.shape)
print(Data.head())
Data.set_index('RowID',inplace=True)
Data.to_csv('NumericData1.csv')
```

Code Snippet for Linear Regression:

```
Data=pd.read_csv('NumericData1.csv')
Data.rename(columns={'Log_ShiipingCost':'Log_ShippingCost'},inplace=True)

corr = Data.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
cmap=sns.diverging_palette(220, 10, as_cmap=True), square=True)
plt.show()

X=Data[['SellingPricePerUnit','PurchasingPricePerUnit','Sales','Quantity','Discount
']]
y=Data['Profit']

print("Score Of KFOLD: ")
linreg = linear_model.LinearRegression(normalize=True)
print(linreg)
```

```
scores = cross_val_score(linreg, X, y, cv=10, scoring='neg_mean_squared_error')

mse= -scores

rmse = np.sqrt(mse)
print('RMSE :', rmse.mean())

deg=range(1,5)
for k in deg:
    poly = PolynomialFeatures(degree=k)
    X_deg = poly.fit_transform(X)
    # predict_ = poly.fit_transform(y)
    linreg = linear_model.LinearRegression()
    scores = cross_val_score(linreg, X_deg, y, cv=10,
scoring='neg_mean_squared_error')

    mse = -scores

    rmse = np.sqrt(mse)

    print('RMSE for polynomial',k,' :', rmse.mean())

name=('Linear','Polynomial-1','Polynomial-2','Polynomial-3','Polynomial-4')
X_axis=np.arange(len(name))
Y_axis=[98.76,98.76, 1.50,0.34,105.80]
plt.bar(X_axis,Y_axis,align='center',width=.4)
plt.xticks(X_axis,name)
plt.xlabel('Algorithms')
plt.ylabel('RMSE')
plt.title("Performance graph of regression Algorithms(Profit prediction)")
plt.show()
```

Code Snippet for implementing K Nearest Neighbor:

```
Data=pd.read_csv('NumericData.csv')

sns.boxplot(x=Data['Discount'])
plt.show()
sns.boxplot(x=Data['CategoryID'])
plt.show()

X=Data[['Discount','DateDif','CategoryID']]
y=Data['OrderPriorityID']

scores = cross_val_score(logreg, X, y, cv=10, scoring='accuracy')
print(scores.mean())

k_range = list(range(500,600))
k_scores = []
for k in k_range:
     knn = KNeighborsClassifier(n_neighbors=k)
     scores = cross_val_score(knn, X, y, cv=10, scoring='accuracy')
     k_scores.append(scores.mean())
print(k_scores)

plt.plot(k_range,k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy(Order Priority)')
plt.show()
```

```
name=('KNN with K=595','Logistic')
X_axis=np.arange(len(name))
Y_axis=[61.57,62.73]
plt.bar(X_axis,Y_axis,align='center',width=.2)
plt.xticks(X_axis,name)
plt.xlabel('Algorithms')
plt.ylabel('Accuracy %')
plt.title("Performance graph of Classification Algorithms(Order Priority
prediction)")
plt.show()
```

Code Snippet for implementing Shipping cost prediction:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from scipy.constants.constants import alpha
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Lasso
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Imputer
import scipy
from scipy.stats.stats import pearsonr
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
import seaborn as sns

Data=pd.read_csv('DS2.csv',usecols=['Quantity ordered
new','Discount','Sales','Profit','Unit Price','Product Base Margin','Shipping
Cost','Log_Sales','Log_ShiipingCost'])
Data=Data=Data[Data['Product Base Margin']>0]
print(Data.head())

print('Shipping Cost prediction :')
X=Data[['Sales','Product Base Margin']]
y=Data['Shipping Cost']


print("Score Of KFOLD: ")
linreg = linear_model.LinearRegression()
print(linreg)
scores = cross_val_score(linreg, X, y, cv=10, scoring='neg_mean_squared_error')

mse= -scores

rmse = np.sqrt(mse)
print('RMSE :', rmse.mean())

deg=range(1,8)
for k in deg:
    poly = PolynomialFeatures(degree=k)
    X_deg = poly.fit_transform(X)
    # predict_ = poly.fit_transform(y)
    linreg = linear_model.LinearRegression()
    scores = cross_val_score(linreg, X_deg, y, cv=10,
scoring='neg_mean_squared_error')

    mse = -scores
```

```
    rmse = np.sqrt(mse)

    print('RMSE for polynomial',k,' :', rmse.mean())

#Shipping Cost pred(Variable tranformation)
print('Shipping Cost pred(Variable tranformation)')

X=Data[['Log_Sales','Product Base Margin']]
y=Data['Shipping Cost']

print("Score Of KFOLD: ")
linreg = linear_model.LinearRegression()
print(linreg)
scores = cross_val_score(linreg, X, y, cv=10, scoring='neg_mean_squared_error')

mse= -scores

rmse = np.sqrt(mse)
print('RMSE :', rmse.mean())

deg=range(1,8)
for k in deg:
    poly = PolynomialFeatures(degree=k)
    X_deg = poly.fit_transform(X)
    # predict_ = poly.fit_transform(y)
    linreg = linear_model.LinearRegression()
    scores = cross_val_score(linreg, X_deg, y, cv=10,
scoring='neg_mean_squared_error')

    mse = -scores

    rmse = np.sqrt(mse)

    print('RMSE for polynomial',k,' :', rmse.mean())



#Outlier removal


sns.boxplot(x=Data['Log_Sales'])
plt.show()
sns.boxplot(x=Data['Product Base Margin'])
plt.show()
print(Data.shape)
Data=Data[Data.Log_Sales<10.3783]
Data=Data[Data.Log_Sales>0.489301]
print(Data.shape)

sns.boxplot(x=Data['Log_Sales'])
plt.show()
sns.boxplot(x=Data['Product Base Margin'])
plt.show()

X=Data[['Log_Sales','Product Base Margin']]
y=Data['Shipping Cost']

print("Score Of KFOLD: ")
linreg = linear_model.LinearRegression()
print(linreg)
scores = cross_val_score(linreg, X, y, cv=10, scoring='neg_mean_squared_error')

mse= -scores

rmse = np.sqrt(mse)
print('RMSE :', rmse.mean())
```

```
deg=range(1,8)
for k in deg:
    poly = PolynomialFeatures(degree=k)
    X_deg = poly.fit_transform(X)
    # predict_ = poly.fit_transform(y)
    linreg = linear_model.LinearRegression()
    scores = cross_val_score(linreg, X_deg, y, cv=10,
scoring='neg_mean_squared_error')

    mse = -scores

    rmse = np.sqrt(mse)

    print('RMSE for polynomial',k,' :', rmse.mean())
```