Master of Science in Data Science & Analytics Thesis

Cross-selling Recommendation and Personalized Advertisement Using Spark

Author:

Alok Kumar Singh

Student Number: 19250990

Supervisor:

Dr. Dapeng Dong

A thesis submitted in fulfillment of the requirements for the degree of MSc. in Data Science and Analytics 2019-2020 in the

Department of Computer Science Maynooth University

August 10, 2020

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study as part of MSc. Data Science and Analytics, is entirely my own work and has not been taken from the work of others and to extent that such work has been cited and acknowledged within the text of my work.

Name: Alok Kumar Singh Date: 10-Aug-2020

Abstract

Cross-Selling is an old and valuable technique used by companies to increase sales size and to transform single product buyers into multi-product ones. Online shopping has been around us for more than two decades. Many e-commerce companies have been very successful, for example, Amazon, Alibaba, eBay, etc. Currently there are around 227.5 million digital shoppers in US. In order to improve online shopping experience and to increase the sales revenue, e-commerce companies often conduct customer behaviors analysis to discover potential buyers as of a product, from this customer behavior analysis they can design a recommendation system to recommend more items to the users to increase sales.

A recommender system is subclass of information filtering system that predicts the rating or preferences of an user for a given item. Commonly recommender systems are used to improve customer experience through personalized recommendations based on prior reviews of the items by the customer. These system tracks different sorts of customer behavior, such as purchase history, viewing habits and browsing activity, in order to model customer preferences. In this we might not have any direct input from the users regarding their preferences; rather we may have to extract features based on the user behavior. In particular, we lack substantial evidence on which products consumer have an aversion to. In this research we consider unique properties of implicit feedback datasets. We propose considering the data as an indication of how frequently the user is reviewing the item with what ratings; for that we will calculate the total number of reviews a reviewer is giving to any particular item. We will be taking the total count of reviews by any particular reviewer for any item and consider that count as one of our features for clustering.

In this work the proposed Cross-Selling techniques is presented in the spectrum of e-commerce aimed at raising the value of a single sales transaction. Various Big Data analytics technologies are used to conduct customer behaviors analysis, in terms of personalized cross-product buying and selling. The proposed cross-selling analysis techniques have been evaluated intensively and comprehensively using real –world product sale transactions.

Acknowledgements

Not all professional do their work by themselves. Although they can be as prolific or as adapt in their respective fields, they will still need assistance one way or another. I have taken efforts in this project. However, it would not have been possible without the kind support and help of my friends and my supervisor (Dr. Dapeng Dong). I would like to extend my sincere thanks to all of them.

I am highly indebted to Dr. Dapeng Dong for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing this project. Throughout the duration of the project his approachability during this pandemic situation made it remarkably easy for me to work and seek his inputs. Without his informative suggestions and critical analysis, this project could not have done. I would also like to thank stack-overflow as it also helped me a lot to learn new things and helped me to fix critical issues while in the development phase.

Last but not the least I would like to thank ICHCE (Irish Centre for High-End Computing) for their system where I was able to perform my heavy computations with big dataset. Working on cluster was another big task for me but their support team helped me a lot and they were approachable at all the time. I would like to thank from the bottom of my heart for all their help and support not only for the system but also for technical support.

Table of Contents

List of Figures

Chapter 1 – Introduction

1.1 Introduction

Recommendation systems are becoming more and more important to many industries as they have been demonstrated to be one of the main drivers for increasing company revenues. As a proof of the importance of recommender system, we can mention that, a few years ago, Netflix organized challenges ("Netflix Prize"), where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1 million U.S dollars for the winners [12]. E-commerce companies to online advertisement organizations, recommender systems are today playing an inevitable role in our daily life.

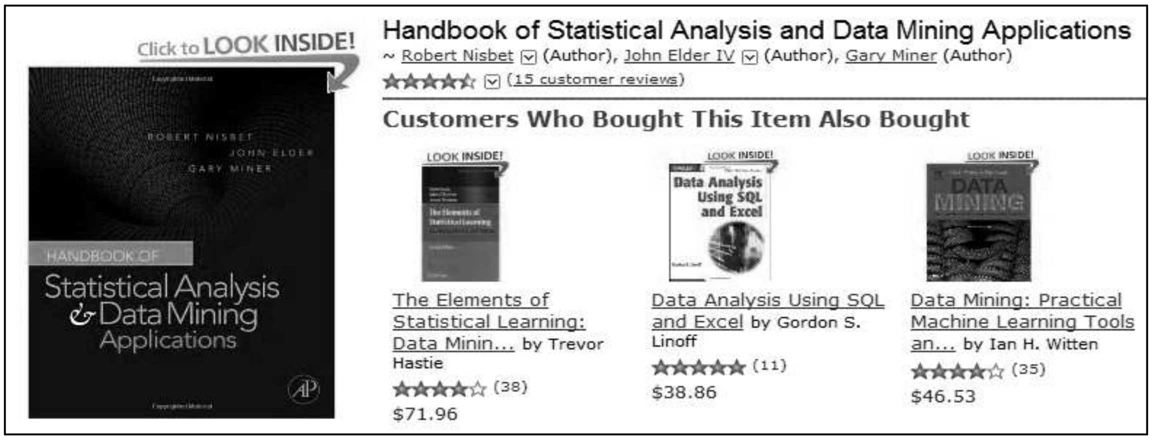
In the changing marketplace in which the economy relies heavily in communication and information technology, the excellency of information and the speed of processing are the key factors for gaining a competitive edge over others. The basis for the effective operations is the implementation of balanced strategies.

Recent years we have witnessed a tremendous growth in the development of e-commerce on the Internet. For example the e-commerce sector in Poland has been developing very rapidly. According to a report from **Gemius**, 73% of internet users shop online and 11 billion PLN was spent on electronic purchase in Poland. The report of Gemius was built on the sample of 1544 internet users aged 15 or above. Data was collected from March 23-30, 2020 [9].

As from Gemius report for Poland we can see that online shopping is growing rapidly and how different mediums (like improving internet services and educating people about benefits of e-commerce) can help in this.

Cross-selling is the action or practice of recommending or selling an additional product or service to an existing customer. The aim of cross-selling is to increase the customer 's revenue and to protect the customer relationship. E-commerce 's dynamic evolution is widely noticed around the world , especially in developing countries. The main factor in the development of e-commerce is technology advancement, which concerns the integration of the business with technology platforms.

According to Fortune Magazine, Amazon is ranked second-most-loved retailer in the world and a major reason for this is the cross-selling of products. There are several factors for the success of the Amazon; one pragmatic and successful strategy is that it motivates loyal customers to buy more products. Amazon typically does not announce its tactics, but CEO Jeff Bezos announced in 2006 that 35 percent of revenues are a direct result of cross-sales [20].



Handbook of Statistical Analysis and Data Mining Applications
~ Robert Nisbet (Author), John Elder IV (Author), Gary Miner (Author)
(15 customer reviews)

Customers Who Bought This Item Also Bought

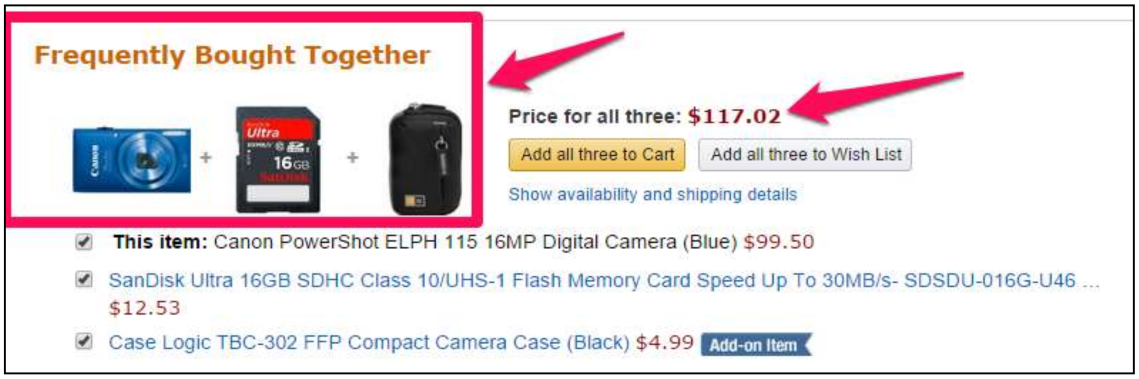
The Elements of Statistical Learning: Data Minin... by Trevor Hastie
(38)
$71.96

Data Analysis Using SQL and Excel by Gordon S. Linoff
(11)
$38.86

Data Mining: Practical Machine Learning Tools an... by Ian H. Witten
(35)
$46.53

Figure 1.1 Example for Cross Selling of books (Source: amazon.com)

Cross-selling is most often described as a competitive advantage for an established company and as a source of synergies supporting an acquisition. The premise behind cross-selling is to grab broad consumer market share by appealing to more customer needs. It's also used in e-commerce online sales like *amazon.com* or *flipkart.com*. As shown in Figure1.1, we can see that while buying one book Amazon suggests other books also in the name of “Customers Who Bought This Item Also Bought This”. This is one type of Cross-selling where Amazon is suggesting based on the items.



Price for all three: $117.02

Add all three to Cart | Add all three to Wish List

Show availability and shipping details

- ✓ **This item:** Canon PowerShot ELPH 115 16MP Digital Camera (Blue) $99.50
- ✓ SanDisk Ultra 16GB SDHC Class 10/UHS-1 Flash Memory Card Speed Up To 30MB/s- SDSDU-016G-U46 ... $12.53
- ✓ Case Logic TBC-302 FFP Compact Camera Case (Black) $4.99 Add-on Item

Figure1.2 Cross-selling of items

In Figure1.2, this is also a technique of cross-selling used by Amazon for increasing the profits. In the above example if you go Amazon.com and select the Canon Powershot Elph digital camera to buy, Amazon will automatically show you other items (a flash memory card and compact camera case) that are "frequently bought together" and as we can see that camera only costs $99.5 but total price of the combination is $117.02. Its human tendency that we all want a a best deal. A savvy online retailer who have turned behavioral economist can help e-commerce companies to achieve more sales in this process.

Research from Harvard Business School showed that bundling products enticed some consumers to buy earlier, but only when buying the bundle is optional [19] as in Figure1.2, we can clearly see that there is a checkbox for the user to either buy the bundle or only buy camera. There's a reason why bundles work – it's all about value. When two or more product are bundled together and sold as one, than there are chances of increase in the perceived value of the bundle as well as the average order value of a particular product.

Travel sites like are very good at cross-selling. If anyone makes a travel plan, they can get hotel recommendations, rental cars and other services related to travel. Bankers do that too-they give you the opportunity to sign up for an extra credit card when you open a checking account. A successful cross-selling will not only encourage consumers to spend more money, but will also make their lives a little easier.

1.2 Objective and Work Done

The main objective of this project is to evaluate several widely used algorithms for item recommendation using Amazon review dataset. We will be using three algorithms to recommend the items from the pool, including K-means, ALS and Naïve Bayes algorithm. For K-means algorithm we will be doing the feature extraction and dividing the customer and items into the clusters and then based on these clusters we will use Naïve Bayes to predict whether the recommended products from the cluster is good or bad for customer and based on this prediction we will do the recommendation, for ALS and Naïve Bayes algorithm we will be using average rating given by the user to various items and the recommendation is based on that only. We will be discussing about these algorithms and how we have used these in a greater details in following chapters.

1.3 Sections Covered in Thesis

In this thesis first we have discussed the introduction of how recommendation systems are used by the companies to earn more profits from the customer and customer retention also. In Chapter 2 we will be doing some literature and technical review of how this works in the real world and we also discuss about cross-selling and up-selling in details and we will also discuss about the technical tools that we are going to use in our research. We will be discussing a little about spark and how it works with big dataset, we will also be discussing the computation system used in the experiments provided by the Irish Center for High-End Computing (ICHEC).

In Chapter 3 we will be discussing about methodology and problems faced during implementing all the algorithms and working with the Big Data. We will be discussing about the design and implementation of the research in this chapter. In Chapter 4 we will be discussing about the dataset (Amazon review Dataset) that we have used. In Chapter 5 we will be discussing about spark and its configuration in details as just by having spark into your system doesn't means you are good to go with any computations. We will be discussing how we can make our spark application work on multiple nodes and how to make a master and slave nodes in spark and we will also see how we can define a spark-session.

In Chapter 6 we will see how we can use classification algorithm with collaborative filtering for efficient recommendation we will also discuss one more algorithm for recommendation, ALS, which uses matrix factorization for better recommendation.

In Chapter 7 which will be our final chapter we will discuss about the future work and conclusion of the work presented in this thesis. We will compare the methods we have used for recommendation.

Chapter 2 – Literature and Technical Review

2.1 Literature Review

Companies and Customer Relationship (CRM) Management specialist have access to rich data on customer behavior. The challenging task is to effectively use this data in CRM process and selection of appropriate data analytics techniques. Data analytics techniques help sales personal to find hidden patterns from customer purchase histories data.

In the era of rapidly changing, globalized economies and highly competitive markets companies need a transformation from a product-centric to a more customer-centric focus. Customers also welcome personalized products and services. The ability to generate useful information from data and to recommend personalized products for customer is essential for CRM specialists. Personalized products for customer can be made by using data mining techniques to find shopping patterns. Commonly age, gender, and geographic characteristics of the customer are used for the customer behavior analysis. Data analytics supported by CRM can be used throughout the organization from forecasting customer behavior and purchasing patterns to identifying trends in sales activities.

CRM is a cross-functional process for achieving a continuous dialogue with customers, across all of their contact and access points, with personalized perks to most valuable customers, to increase customer retention and the effectiveness of marketing initiatives. Typically, there are four phases of the customer management lifecycle: Customer Identification, Customer Attraction, Customer Retention, and Customer Development.

Cross-selling and up-selling are of great importance in the context of customer relationship management, as they give reasons for effective advertising, promotion and loyal campaigning. It also helps to reduce the risk of losing already existing customers. Cross-selling and up-selling are well-known marketing methods that aim to increase the value of a single sales transaction and increase customer confidence.

2.1.1 Up-selling

Up-selling is term which means asking the customer to buy a more of related products or of something else Up-selling means moving up to more expensive versions of what customer is already consider purchasing. This method is mostly used by the companies before the customer actual purchase of the item but after the customer selects the product that they want to buy.

For example, as shown in Figure2.1 a customer was going to buy a cheap television then the seller might offer a television which is a little more expensive than the customer choice but with extra features.

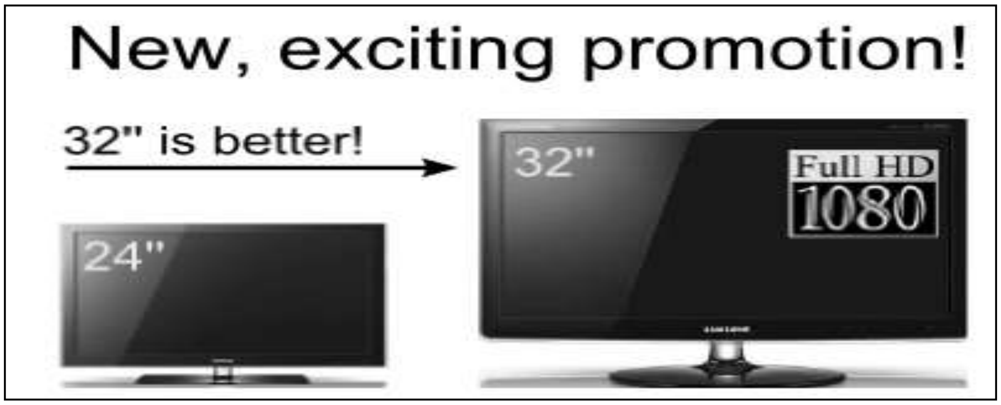


Figure 2.1An Example of Up-Selling

2.1.2 Cross-selling

If it comes to growing sales per customer, cross-selling is one of the best and most valuable strategies for any company. Cross-selling is often seen as a source of competitive advantage for an established company and as a source of synergies which justify an acquisition. This is the strategy by which businesses have incentives and enticing deals for the seller to purchase additional products. Cross-Selling also includes offering the customer products which in some way complement the original purchase. The concept behind cross-selling is to gain a significant share of the consumer market by fulfilling more customer requirement.

For example, if a customer was going to buy a Smartphone then the seller might suggest an earphone to the customer.

Within the context of CRM, cross-selling has become a valuable strategy for customer development. In general, it is considered that it is easy to serve an existing customer than to make a new customer [25]. Cross-selling leads to a broader scope for the CRM, by not only increasing the profit for customer but also for organization.

The biggest limitation of exaggerating cross-selling efforts is getting the customer excessively-reaching. Reaching the consumer with cross-selling deals every time will cause the customers to ignore these efforts, thus making customers reluctant to receive any further cross-selling offers. Customer might get annoyed then cross-selling produces the opposite of its actual objective, leading to erosion of the customers. This is the reason, why most of the companies don't tend to reach the customer and wait for the customer to contact them, with their issues, first they resolve their issue and then they cross-sell. As if a customer is calling for any issue and the issue is resolved then there are more chances of cross-selling.

In 2016, Wells Fargo [18] was charged with a $185 million fine for opening bank and credit card account for thousands of customers without their knowledge or consent. As a result FINRA the independent regulatory body for US securities firms, launched an investigation of cross-selling practices at 14 broker- dealers, with a spokeswoman stated that: "In light of recent issues related to cross-selling, FINRA is focused on the nature and scope of broker- dealers cross-selling activities and whether they are adequately supervising these activities by their registered employees to protect investors". Therefore cross-selling must be carried out with great care, in order to offer the right product to the right customer at the right time.

2.1.3 Analytical Tools for Cross-Selling

The analytical tools that allow cross-selling in the context of CRM can be divided into two main groups: ***Pattern Analysis*** and ***Collaborative Filtering*** acquisitions. The main objective of an analysis of the purchasing trend is to determine the next logical phase for the consumer in terms of product purchase based on the pattern of previous purchase and other customer purchase experience.

For instance, a businessman seeking a Personal Digital Assistant (PDA) might need a carrying case next, followed by additional memory, software, etc.

While collaborative filtering examines the sequence of associations between customer-wide purchases, it identifies suggestions for other items which can be recommended with the bought one.

For example if a book, was added to the shopping cart, Amazon might suggest other books purchased by customers who has already bought the same book.

2.2 Previous Research

Numerous researches concentrate on a specific variable sales forecasting, taking into account the information based on parametric methods in which the random distribution type is given. For data of a normal demand, ***R.A.Fisher*** *[5]* has Built a mean and standard deviation estimator by maximizing the likelihood function where two specially predetermined tables were added. ***Cohen****[6]* devices a new formulae to strengthen the Fisher method of optimizing the likelihood function, where the special tables used by Fisher are eliminated. After Cohen, ***Gupta*** *[7]* investigated the same problem and presented a better linear unbiased estimator then Cohen and Fisher.

Nahmias *[8]* established a parametric methodology to estimate normal demand, and validated the censored sample approach. ***Berk*** **[16]** with his research has estimated demand with censored data of negative binomial, gamma, Poisson, and normal distributions through Bayesian updates, approximately posterior distribution. ***Jain*** **[17]** in his research modeled the arrival of cumulative demands as a stochastic process with unknown parameters. By observing sales and stock-out events, which were further simplified by other researchers, the parameter was updated with a Bayes theorem by introducing the stock-out time observations

Ren-Qian Zhang, Qi-Qi Wang, Yi-Ye Zhang, Hai-Tao Zheng and Jie Hu **[3]** they did not consider the uncontrollable loss of sales resulting from stock-outs and the positive externalities of cross-selling. They proposed a method for estimating the normally distributed demand for cross-selling products, based on censored sales data with uncontrollable loss of sales.

The California State Automotive Association has evaluated its database and found that a participant who used roadside assistance within a year is more likely to continue to use the service and is a safer candidate for policies related to cross-selling insurance. But at the other hand, a customer of auto insurance who also enrolled for roadside assistance but never used it in the first year is probably not using the service.

With benefits of cross-selling, there is always a risk for the firm to exaggerate its cross-selling strategy, thereby its customer feels isolated or estranged. However, in most of the cases, customers' reaction to cross-selling is surprisingly positive. To see how the customer behaves with cross-selling, in 2005, Forum Corp. conducted a survey with a sample of 1624 customers worldwide, showing that 88 percent value service reps recommend alternative goods that better serve their needs. 42 per cent of customers surveyed said they often or regularly buy additional services. The explanation for their willingness to consider purchasing additional products / services was current satisfaction purchase and how well the additional services presented to them meet their needs [21].

2.3 Technical Background

MapReduce have been very useful in implementing distributed, large-scale, and data intensive applications. MapReduce is one of the data processing frameworks which deal with big data for distributed computing primarily based on Java programming language. As the name suggest MapReduce applications contains two important tasks Map and reduce.

Map take a set of data and transform it into another collection of basic data where individual elements are further broken down into tuples (Key-Value pair).

Reduce works as an object with the Map output and merges such data tuples into smaller tuple sets. Reduce task is always performed after the Map job.

Cluster computing has become widely popular, in which parallel data computations are executed on clusters of unreliable machines by system. MapReduce pioneered these model users just have to create acyclic data flow graph to pass input data through a set of operators and systems achieve the framework will deal with system scalability and fault tolerance. This programming model for data flow is useful for broad class of applications.

Apache Hadoop MapReduce is a framework for processing big dataset in distributed manner throughout the Hadoop cluster. Hadoop framework uses map and reduce functions, and providing the scheduling, distributed and parallelization services.

MapReduce can be inefficient in some of the cases, for example:

1) Iterative jobs
2) Interactive analysis

Iterative Jobs: Many of the machine learning algorithms applies the same functions repeatedly on the same dataset to optimize the result. In this process each job reloads the data from the disk which causes the performance issue.

Interactive Analysis: Hadoop is often used to run ad-hoc queries on large datasets, through SQL interfaces such as Pig and Hive. A user would be able to load a small dataset into memory across a number of machines in a cluster and query, without affecting the performance, as the data is distributed in different machines, and no single machine has heavy load. With Hadoop each query incurs significant latency (tens of seconds) because it runs as a separate MapReduce job and reads data from disk.

2.3.1 Spark

Most of the system is built around acyclic data flow model (graph in which there is no cycle, or closed path. In other words, it is a path with no repeated vertices excluding the starting and ending vertices) which is not suitable for most of the popular applications. This includes many iterative machine learning algorithms as well as interactive data analysis tools. In these cases, we will be using a framework provided by Apache Spark. Spark retains the scalability and fault tolerance of MapReduce. Spark can be more effective compared to Hadoop alone in iterative machine learning jobs and can be used to query large dataset with a small latency e.g. query 39Gb data in 30 seconds.

Spark is an open source distributed cluster programming framework. Spark provides an interface for programming entire clusters with parallelism and fault tolerance. In Spark codes are written in such a way that it controls the high-level flow of their application and launches various operations in distributed system.

Spark achieves high performance for batch and streaming data, using DAG scheduler (Directed Acyclic Graph). DAG scheduler is a query optimizer which optimizes the query and a physical execution engine.

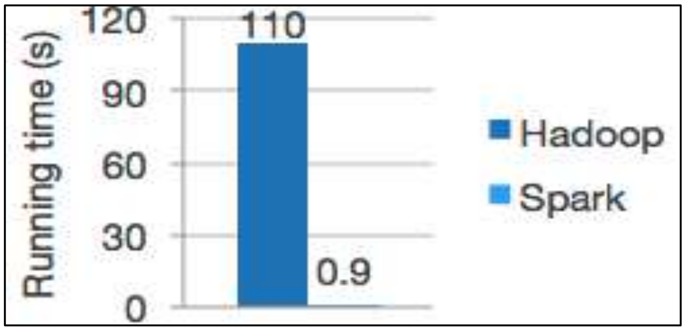


Figure 2.2 logistic regression in Hadoop and spark (source: spark.apache.org)

In Figure 2.2 we can clearly see that spark is around 10 times faster than Hadoop while applying same logistic regression.

Spark was first created using Resilient Distributed Datasets (RDD). RDDs are read only multiset of data items that are distributed over a cluster of machines to perform parallelism, that is maintained in a fault tolerance way. The Dataframe API is a concept on top of the RDD, followed by Dataset API. In Spark 1.x versions the RDD was the primary application for programming interface, but from Spark 2.x release, use of dataset API is encouraged even though the RDDs are not deprecated.

Resilient Distributed Dataset (RDD): RDD is a simple spark data structure. It is an immutable distributed collection of objects. Each RDD in dataset is divided into logical partitions, which may be computed on different node of the cluster. RDDs can contain any type of objects from Python, Java, or Scala, including user defined classes. RDD is a read-only partitioned collection of records and also provides fault-tolerant collection of elements.

Datasets: A dataset is distributed collection of data. It is a new interface added in spark 1.6 which provides the benefits of RDDs (strong typing, ability to use powerful lambda functions) with the benefits of spark SQL's optimized execution engine. It can be constructed from JVM objects and then can be manipulated using functional transformations (map, flatMap, filter, etc.). *(Source: spark.apache.org)*

Dataframe: A dataframe is a type of dataset which is organized into named columns. It is conceptually equivalent to a table in relational database or a dataframe in R/Python, but with richer optimizations. Dataframe can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs. *(Source: spark.apache.org)*

The Dataset API for spark is only supported in Scala and Java. Python does not have dataset API. But Dataframe API for spark is supported in Scala, Java, Python and R. Due to this drawback of dataset we will use spark Dataframe for our cross-selling and recommendation system for all the algorithms (K-Means, ALS, and Naïve Bayes) that we are going to use in this project. We will be discussing more about Spark in next chapter how to configure and what all issues faced while using it in distributed system.

Chapter 3 - The Solution

3.1 Methodology

To work with a *big* dataset we would need a system which can handle the heavy computation and processing of the dataset. For this we were provided with ICHEC (Irish Center for High-End Computing) server which is having great configuration to perform heavy computation.

ICHEC delivers complex compute solutions to Irish Higher Education Institutions (HEIs), Enterprise and the public sector on behalf of the state. ICHECs National High-Performance Computing Service offers academic researchers access to HPC (High Performance Computing) computing resources.

An HPC is more than just supercomputers; it is a cluster which support the numerical simulations and data analysis required by scientific research.

Kay is ICHEC's primary supercomputer and Ireland national supercomputer for academic researchers. Kay is comprised of a number of components:

Cluster (ProdQ, DevQ and LongQ nodes): It consists of 336 nodes in total where each node has 2 x 20 - core 2.4 GHz Intel Xeon Gold 6148 (Skylake) processor, 192 GB of RAM, a 400GB local SSD for scratch space and a 100GB OmniPath network adaptor. This partition has 13440 cores and 63TB of distributed memory.

GPU (GpuQ node): This partition consists of 16 nodes with the same specifications as above plus 2 x NVIDIA Tesla V100 16GB PCle GPUs on each node. Each GPU has 5120 CUDA cores and 640 Tensor cores.

Phi (PhiQ node): This partition also consists of 16 nodes, each containing 1x self-hosted Intel Xeon Phi Processor 7210 (Knights Landing or KNL architecture) with 64 cores at 1.3 GHz, 192 GB RAM and a 400 GB local SSD for scratch space.

High Memory (ShmemQ node): This is a set of 6 nodes each containing 1.5 TB of RAM, 2 x 20-core 2.4 GHz Intel Xeon Gold 6148 (Skylake) processors and 1 TB of dedicated local SSD for scratch storage.

Service & Storage: A set of service and administrative nodes to provide user login, batch scheduling, management, networking, etc. Storage is provided via Luster file systems on a high-performance DDN SFA14k system with 1 PB of capacity.

In next section we outline the system requirements for Big Data analytics using spark.

3.1.1 System Requirement

The Spark provides a python API (PySpark) which exposes the spark programming model to python. PySpark requires python 2.6 or higher versions. PySpark applications are executed using a standard CPython interpreter in order to support python modules that use the C extensions.

Storage Systems: Since most spark jobs will likely read input data from an external storage system it is important to place it as close to the system as possible. The data should be placed on the same machine in which we are running spark. If it is possible, run spark on the same nodes as Hadoop Distributed File System (HDFS), but, if not possible, run spark on different node in the same local- area network as HDFS.

Local Disk: While Spark performs a lot of its computation in memory, it still uses the local disk to store data that doesn't fit into Random Access Memory (RAM) according to spark.apache.org 4-8 executors per node are recommended.

Memory: It can run well with memory anywhere between 8GB to 100GB per machine.

CPU: Spark works well to tens of CPU per machine because it performs minimal sharing between threads. One should at least provision 8 to 16 cores per executor machine.

3.2 Design and Implementation

Data related to Amazon Reviews can be easily downloaded from the Amazon API. To perform our algorithms on big dataset which will lead to heavy computations, we would need python with spark to work. For python with spark ICHEC helped us by providing their system with all the dependencies installed in it. We just have to create our virtual environment and install the dependencies in the system using pip install. More details of the required software will be discussed in the Technical Issues section.

3.2.1 Machine Learning Algorithm

Applying ML algorithm needs few steps to be followed:

1) Data Collection and Preparation
2) Feature selection and feature engineering
3) Select the algorithm to be applied and train the model
4) Evaluate the Model
5) Model tweaking, regularization and hyper parameter tuning in this step we try to make best model out of good model which we trained in 3rd step.

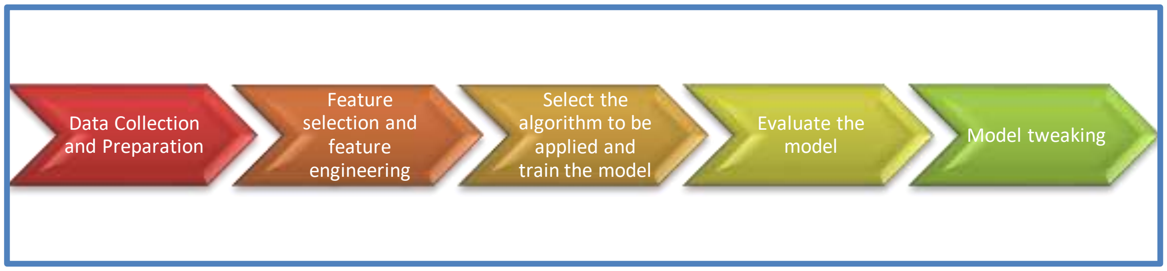

Data Collection and Preparation
Feature selection and feature engineering
Select the algorithm to be applied and train the model
Evaluate the model
Model tweaking

Figure 3.1 Steps involved in ML

Among these steps first and second steps are the very crucial and important steps since if the data is not clean enough to be processed then it will produce ambiguous models.

The workflow in recommending the items for customer from the dataset involves multiple step.

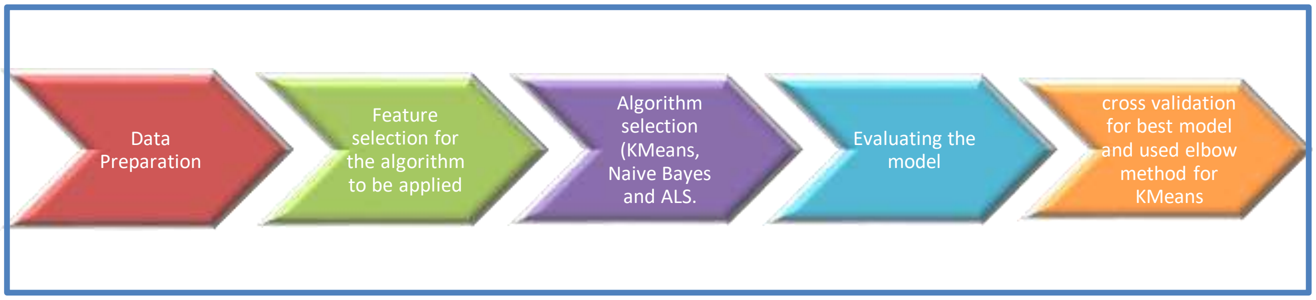

Data Preparation
Feature selection for the algorithm to be applied
Algorithm selection (KMeans, Naive Bayes and ALS.
Evaluating the model
cross validation for best model and used elbow method for KMeans

Figure 3.2Workflow for recommendation system.

3.3 Technical Issues Encountered and Their Solutions

The technical issues were mostly related to the supercomputer we were provided. After getting the credential for accessing Kay, first thing I had to face was that there are two nodes login node and compute node with various different partitions for compute node. In login node, we can use internet and this node is only used for downloading something from internet, it can be useful for small-scale proof of concept experiments. Then comes compute node with various partitions of node with different wall time, but selecting a perfect node need a research for their specification, as if you use more RAM or less RAM in both the cases, we will encounter error related to memory. Then for using PySpark in Jupyter we have to create a virtual environment as we have to install different libraries according to our requirement, and versions of the library should also be considered.

PySpark doesn’t work well with python 3.8 version and when you install python in virtual environment without mentioning the version it will install latest version of python. So after some time when I realized that, I changed the python version to 3.4 and then installed PySpark 2.4.5 it worked well, but, when I have to use matplotlib library for plotting then it was giving me error to change the python version to more than 3.4 as matplotlib won’t work with python 3.4 so after some research I found that python 3.5, PySpark 2.4.5 and matplotlib all can work well together so I changed the version again from 3.4 to 3.5 for python and installed all the other dependencies.

This was not the end of my problems it was just a starting of new problem running experiments with the full-size of the dataset as this involves heavy computation. We have to configure Spark to work with big dataset, as having spark doesn’t means it can handle everything. If we won’t configure spark it will work with default values assigned to its variables which might not work with. Configuring Spark will be discussed in detail in Chapter 5.

Chapter 4 Data Exploration and Analysis

4.1 Overview

Amazon product review and ratings are a very important business. Customers on any e-commerce website often make purchasing decisions based on those reviews and a single bad review can cause a potential buyer to reconsider the purchase. Amazon customer review is one of Amazon's best strategies towards customer retention and earning more profit. Millions of Amazon customers have contributed over a hundred million reviews to express opinions and describe their product experiences on the Amazon.com website over a period of more than two decades since the first review in 1995. This makes Amazon customer review a rich source of information for researchers in the Natural Language Processing (NLP) and Machine Learning (ML) fields. We will be using this dataset provided by Amazon and introduced in **[23]** to build our recommender system using ML and spark.

4.2 Basic Statistics and Analysis of Dataset

The dataset consists, reviews of customers starting from May 1996 to September 2018, for almost 23 years. This dataset consists total of 230.13 million ratings from 43.24 million users million users for 14.89 million items.

Table 4.1 will show the structure of dataset that we are going to use:

Table 4.1 Dataset Structure

Name of Columns	Description	Type
Asin	ID of the product eg:- 0992916305	String
Image	Link to show image	String
Overall	Ratings provided to Items	Double
reviewText	Reviews given by user to items	String
reviewTime	Time when user reviewed the item	String
reviewerID	ID of reviewer eg:- A2V9BG2MDQVCYX	String

reviewerName	Name of Reviewer eg:- Alex G	String
Summary	Summary of the review given by the user	String
unixReviewTime	Unix format of time when reviewer reviewed the item	Bigint
Verified	Gives detail of verified users	Boolean
Vote	Vote by users who find the review helpful	String

Sample dataset one row from the Amazon_Fashion dataset:-

{"overall": 3.0, "verified": true, "reviewTime": "04 13, 2013", "reviewerID": "AKS3GULZE0HFC", "asin": "B00007GDFV", "style": {"Color:": " Black"}, "reviewerName": "M. Waltman", "reviewText": "I had used my last el-cheapo fake leather cigarette case for SEVEN YEARS. It still closed completely but the plastic made to look like leather was literally falling off, so it was time for a new one. Cigarette cases for kings size cigs are not easy to come by these days I discovered, but I was thrilled to find this one on Amazon. It was a great price, REAL LEATHER, and even had the cool zipper pouch on the back. I was so excited to get my case and toss that other one! Well, within THREE DAYS one of the gold clasps literally broke off! I couldn't believe it! I tried to super glue it back on and was not successful. so, I still use the case but it doesn't close securely. I was very disappointed that my $3.00 plastic one lasted 7 years and this real nice leather one lasted 3 days!! But I still love the zipper pouch on the back, it's great for the spare key to my car because I will not go ANYWHERE without my cigarettes!", "summary": "Top Clasp Broke Within 3 days!", "unixReviewTime": 1365811200}

The dataset consists of various categories of Amazon products, for example: the Amazon_Fashion contains the details of the fashion related product; the Books dataset have reviews related to Books; the Clothing_shoes_and_Jewelry dataset contains reviews related to clothes, shoes and jewelry related reviews that they have either purchased and used it or just reviewed on their past experiences; the Musical_Instruments dataset contains the reviews on musical instruments such as guitar, piano or flute etc. There are total of 30 different categories of Amazon products. This complete dataset is about 150 GB in size containing reviews on 14.89 million items 30 categories.

Ratings provided by the customer for multiple items in the above 30 categories are shown in Figure 4.1:

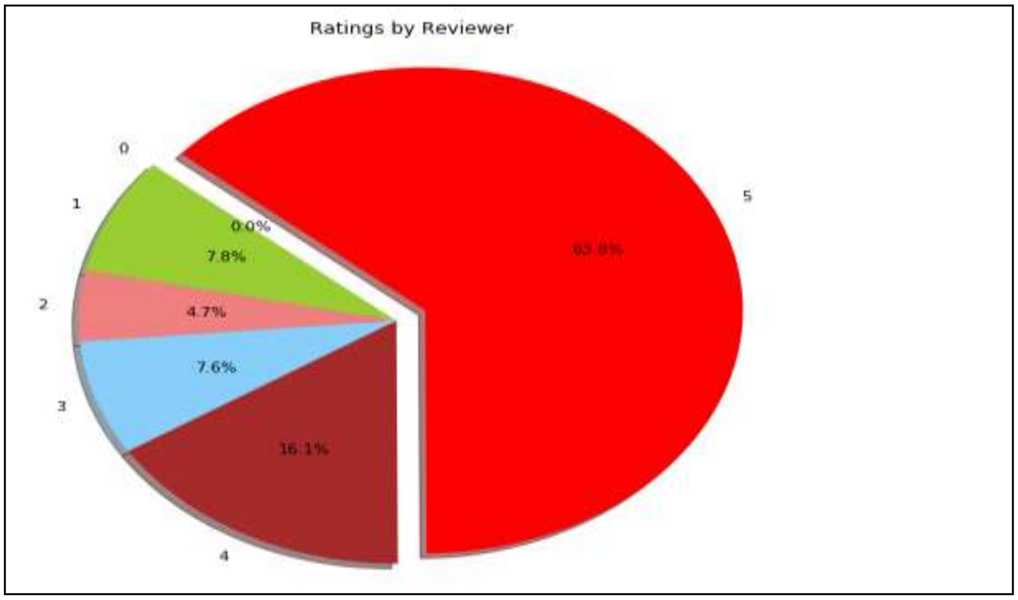


Figure 4.1Rating by Reviewer

In Figure 4.1 we can see that most of the items are rated as 5, more than 50% of items rated by the customers are at 5 and then followed by 4. This shows that there is higher proportion of customer giving only 5-star reviews.

4.3 Exploring the Dataset

As discussed above the dataset contains customer reviews across 23 years with millions of users reviewing millions of items. We will also explore how review system changed for Amazon with time. Table 4.2 shows some basic statistics about the dataset.

Table 4.2 Basic Statistics

Items	**Stats**
Total Reviews	230139802
Total Items Reviewed	14894121
Total Users	43249276
Total Verified Users	35177435
Total Un-Verified Users	8071841
First Review	May 1996
Last Review	September 2018

Amazon started its review system in 1996 and started collecting the review of the customer to make their life easy and recommend items based on their previous reviews by learning the behavior of their customer towards the items that they are using.

In 1996 when Amazon started this review system, most of the users were not giving their reviews but once the Amazon started gaining their trust, that they will deliver the best based on their reviews, then reviews gradually started increasing. Table 4.3 shows how the reviews increased with respect to time.

Table 4.3 Reviews w.r.t Time

Year	**Total Reviews**
1996	39
1997	14022
1998	77719
1999	182357
2000	576227
2001	549400
2002	565073
2003	596360
2004	694864

2005	957778
2006	1063117
2007	1624518
2008	1982794
2009	2487499
2010	2940407
2011	4313497
2012	7851424
2013	19672327
2014	31871589
2015	44228378
2016	50595500
2017	39298027
2018	17996886

As we can see from the Table 4.3 that initially there were very less reviewers giving their reviews on the items only 39 reviews in 1996. In 2000’s or in the last of the decade 2000 customer started giving more reviews and in 2016 it recorded the maximum ratings i.e. 50595500 (50.59 million). This is shown in Fig 4.2.

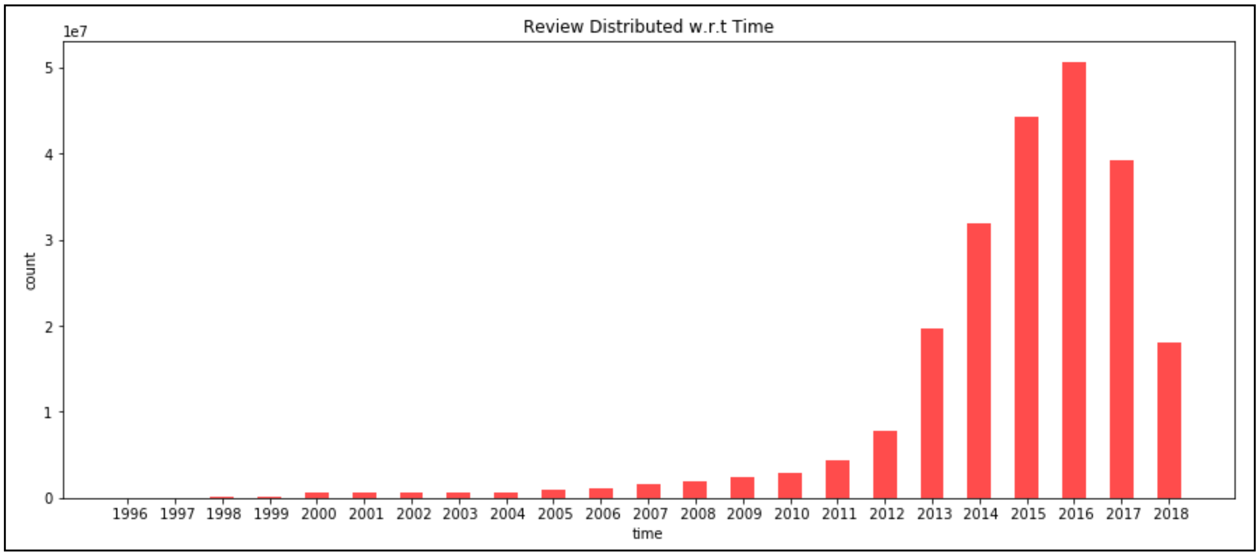


Figure 4.2 Distribution of Reviews w.r.t. Time

Fig 4.2 shows how the number of reviews has gained momentum. We can see the peak in number of reviews in just three or four years. From the plot we can see that after 2010 the review system for Amazon gained popularity and this the only year when revenue for Amazon sky rocketed. According BBC[22] forecast suggesting that revenue could pass $320 billion by the end of year 2020 as shown in Figure 4.3

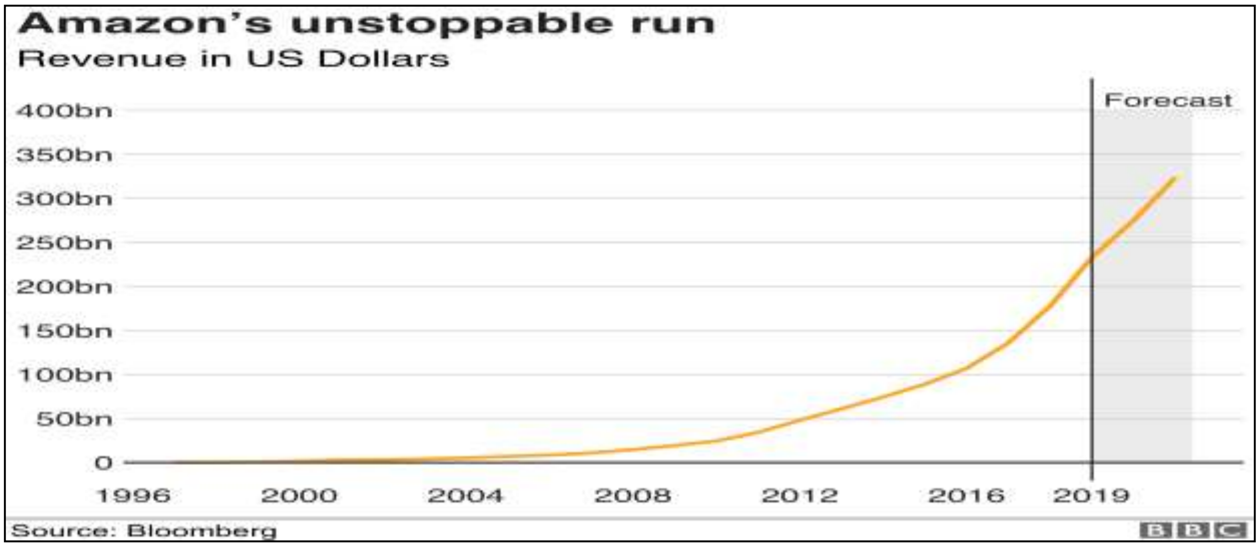


Figure 4.3 Amazon Revenue Growth

Recommender system plays vital role in any e-commerce site success and same goes with Amazon as well. Better recommender system shows how well you know your customer and this helps in building trust between customer and company.

Chapter 5 – Spark Configuration

5.1 Introduction

Apache Spark is a unified analytics engine for processing large scale data. Runs the workloads 100 times faster than any other distributed system. Spark powers a stack of libraries including SQL and Dataframe, Mlib for machine learning, GraphX and Scala Streaming. All these libraries can be combined in single application. (Shown in Figure 5.1)

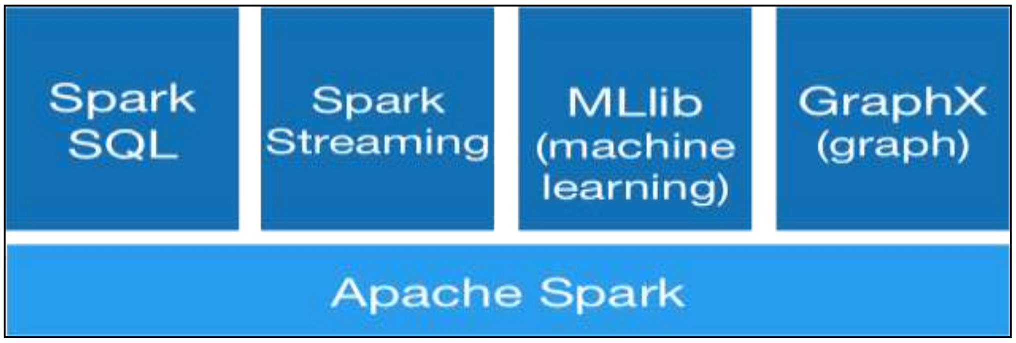


Figure 5.1 Combining libraries in one application

We can run Spark using standalone cluster mode, on Hadoop Yarn, on Mesos, on EC2, or on Kubernets. Data can be accessed in various data source such as HDFS, Alluxio, Apache Cassandra, Apache HBase, Apache Hive etc.

Figure 5.2Data source for Apache Spark

5.2 Spark Configuration

Apache Spark is a very fast cluster computing technology, designed for fast computation. Spark's key feature is in its memory cluster computing which increases an application's processing speed. Spark is not a modified version of the Hadoop framework as opposed to a common misconception, and is not really based on Hadoop alone because it has its own independent cluster management. Hadoop is just one way to get Spark implemented.

For Spark installations there are two basic applications that need to be installed into the system first, that are Java and Scala. Spark can be installed on any operating systems for example, Linux, Windows or macOS. But before installing spark we have to verify that Java and Scala.

To verify for Java we will run a following command to verify java version:

$java –version

If Java is already installed in the system, it will return with its version otherwise we have to first install java before we proceed further.

After Installing or verifying for java we have to verify it for Scala this is one of the major requirements to install Spark. Same as java we can check for the Scala version also and verify that it is installed or not.

$scala –version

If already installed, it will return with Scala version. Otherwise you have to install Scala.

After verifying for Scala and java we can proceed with installing spark. To install Spark you have to install its tar file from the apache spark site. Spark provides two options to install spark from its site that are:

1) Source code
2) Packaged version

If you install source code then that needs to be compiled otherwise errors such as jar file are missing will come whereas in packaged version, spark do all these steps for you and, you just have to use it after installing it to all the nodes where you will be using spark for computation.

Spark is an open source tool this is the reason spark provides the spark code so that if you need to do any changes to the code and make any customization to the code you can do it easily and use it as per the organization business logics. But for research purpose it is advised to use the packaged version of spark until or unless if your research is on spark itself.

After installation extract the tar file using

$ tar xvf spark-xxx.tgz

You can verify the spark installation by running

$ spark-shell

If spark is installed successfully then the output screen will give you access to write Scala code in it.

In my case all the above steps mentioned were done by ICHEC as it was their system and they only have the access to install anything on the nodes.

It needs to be configured to work efficiently and to do heavy computations.

5.2.1 Running Spark with Python

Before running spark applications and using Spark we have to install *PySpark* into our virtual environment that we have created to run our spark application and this can be done using the following command.

$ pip install pyspark

After installing PySpark some configuration files need to be set according to our needs. Spark provides the template for these files in conf directory of spark. These files are easily accessible from */spark/conf/* and the files are *spark-env.sh.template, spark-defaults.conf.template and slaves.template,* and if you wish to update log properties then we can modify *log4j.properties.template* we just need to copy these files in the same location removing template from it and we can start modifying these files. We basically set some of the properties of the global variables for spark with the help of these files. When we run Spark it calls these files to set those variables which we define in these files and what we don't define it will consider their default values only.

Since these files are present at server level in my case ICHEC we don't have the permission to modify these files so in that case we have looked for another option. From spark-2.x spark introduced the concept of spark-session which turns out to be the entry point for any spark application. In Jupyterhub notebook all the spark application starts with spark-session that we define in the code basic syntax for the spark-session is:

spark = SparkSession \.builder \.appName("AmazonCrossSelling") \.getOrCreate()

SparkSession is the entry point for spark applications, the builder is to build the spark session, *appName* is to provide a name for spark application and *getOrCreate()* is to either get the spark session if it is already present and if not then create it. Using .config() we can add configuration for spark like number of executors, total number of cores required (cores are basically the number of CPU per executor), executor memory, number of workers for parallelism, worker memory etc. There is no specific formula to set these configurations we just have to use hit and trial method.

There are three ways by which we can assign values to these global variables. One method is by assigning the values through *conf* files as discussed above, second is by defining in the spark-session and third is by defining in script file when we are running the spark application as a job. Before these global variables, sometimes we need to run our code on multiple nodes (these are basically multiple systems) and before running these on multiple nodes we have to make these nodes as master and slaves. So that all the slaves are connected to its master and run in parallel.

We just have to run two shell files provided by spark to make any node master or slaves these files are *start-master.sh, start-slaves.sh and start-all.sh. start-master.sh* will make the node as a master on which it will run; *start-slaves.sh* will make all the nodes as slaves for this master. We can make master and slave both at single node only then we have to run both the commands at single node only or we can run *start-all.sh* which will run both shell files on single node only and make it master and slave both. These files are available at */spark/sbin/, SPARK_HOME and JAVA_HOME* path also needs to be defined before running these shell files and make sure that you have the complete access of those directory because when we run these files it creates a directory for log in these home directory and write a log file so if you don't have the access to these home directory it will end-up with an error such as permission denied.

After starting master and slaves we should also stop the master and slave otherwise it will make the nodes busy even if it's not using the resources for 60 minutes by default. Obviously, we can set this time something less and can check if the master and slaves are idle for some time so we can kill the process and stop these shell files hardcoded. There is another way also which is preferable to stop these files simply run shell files provided by spark that are *stop-master.sh*, *stop-slaves.sh and stop-all.sh* for example:

```
./spark/start-master.sh
./spark/start-slaves.sh
spark-submit xyz.py
./spark/stop-slaves.sh
./spark/stop-master.sh
```

In this script we will first run the master shell file then slaves shell file and after that we will run our spark submit to run our .py file after .py file execution is complete we will stop master and slaves by running the *stop-slaves.sh and stop-master.sh* files.

After doing all the configuration part for using master and slave we should decide what clustering method we are going to use as spark on the clusters. Spark provides three clustering methods:

1) Hadoop YARN (Yet Another Resource Negotiator): The resource manager in Hadoop and obviously you need Hadoop to be installed into your system to use this.
2) Standalone: A simple cluster manager included with spark that makes it easy to set up cluster.
3) Apache Mesos: A general cluster manager that can also run Hadoop MapReduce and service applications.
4) Kubernetes: an open source system for automating deployment, scaling, and management of centralized applications.

We will be using standalone mode for our research. For monitoring the submitted applications driver program has a web UI, typically on port 4040 that displays information about running tasks, executors, and storage usage. Simply we need to go to *http://<driver-node>:4040* in a web browser to access this UI. Every spark-session launches a Web UI by default on port 4040, that displays information about the application. This includes:

1) A list of scheduler stages and tasks
2) A summary of RDD sizes and memory usages in our case dataframe size.
3) Environmental Information.
4) Information about the running executors.

The above information is only available till the application is running. To view the web UI after the application stopped or gets aborted due to any reason, set *spark.eventLog.enabled* to true before starting the application. This configures Spark to log spark events that encode the information displayed in the UI to persistent storage.

To view the UI we will use spark history server, provided that the application's event log exist. To start the history server, we have to execute the following command:

```
$ ./spark/sbin/start-history-server.sh
```

This creates a web UI at *http://<server-url>:18080* by default, listing incomplete and completed applications and attempts. When using the file system provider class, the base logging directory must be supplied in the *spark.history.fs.logDirectory* configuration options and should contain sub-directories that each represent an application's event logs.

These web UI are very useful in deciding the memory needs to be allocated to the worker and executor and also in deciding the number of instances as there is no method to decide these numbers and just by monitoring the dashboard we can guess values close to perfection where we can use spark at its best and can also decide what level of parallelism we require for our application, so that we don't get any memory related issues.

In our research we won't be able to see any web UI on any ports as to launch this web UI for monitoring spark application we would need internet on the nodes it is running, we will be using compute nodes provided by ICHEC and due to the security reasons they don't provide internet access for compute nodes. We will be using log files generated by spark framework to see what is happening with our spark applications and how the nodes behave with our master node.

Chapter 6 Implementation and Evaluation

6.1 Overview

For our recommender system we will be using three algorithms for recommendation namely: K-means with Naïve Bayes, and we will also use ALS (Alternate Least Square) which is very common for recommendation algorithm provided by spark. To apply these algorithms, we will be using two files of Amazon product review dataset as discussed in Chapter 4 i.e., Amazon_Fashion and All_Beauty.

We will be doing few modifications to our dataset for feature selection in K-means algorithm such as imputing the values for vote column and handling NULL values in different columns, whereas in ALS and Naïve Bayes algorithm we will be using ratings given by the user to various items for recommending the best item. In ALS and Naïve Bayes, we will be converting strings into its indexes as these algorithms don't work well with strings.

6.2 Implementation and Evaluation

In this section we will be going through each algorithm that we have used in detail. Starting with K-means first we will see how K-means algorithm works and how it can, with the help of Naïve Bayes algorithm recommend items to the user, then we will go with ALS and see how this algorithm works.

We will be training our dataset and testing it by its prediction but as we know there is no correct or wrong model for recommendation of items to user, it's always either good or bad model for prediction. There is a famous **George Box** quote ***"Essentially, all models are wrong, but some are useful",*** [26]. So, to find the best model, before training our model we will be doing some modification to our dataset and before testing our model we will be doing cross-validation to find the best model from the option available to us.

6.2.1 Cross-Validation

Usually we divide our dataset into three parts:

Training: Used to train our model

Testing: Used to test the trained model on unknown dataset and check how our model will behave with real world problems.

Validation: During testing some information about test set leaks into the model so we perform a final test on completely unknown dataset.

Cross-validation helps to reduce the need of validation set because you are able to train and test on the same data. In most common cross-validation approach we use part of training set for testing. We do this several times so that each point in the set appears at least once in a test dataset.

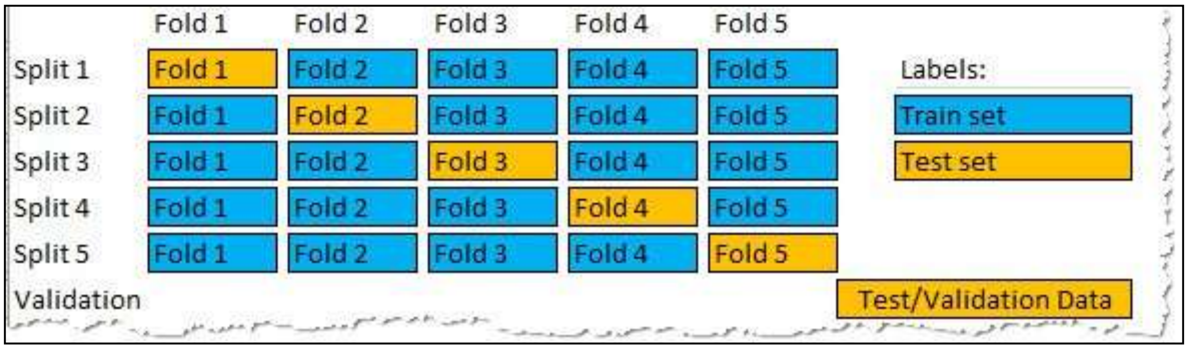


Figure 6.1 Cross-validation (Source:- towardsdatascience.com)

Cross-validation is method of model validation which splits the data in order to obtain the better estimates of real-world model performance and minimizing validation error as shown in Figure 6.1. There are various types of cross-validation; some of widely adopted mechanism includes KFold, ShuffledKfold, and StatifiedKfold. We will be discussing about KFold method which is very popular for cross-validation.

6.2.1.1 KFold Cross-Validation

Before KFold we should understand holdout method in this we simply divide the dataset in two parts train and test based on the size of the dataset usually we take 70-30 random split of dataset, but since we are working with big dataset, we will take 60-40. After splitting we check for the performance of the dataset. Now coming to KFold how it uses holdout method, In KFold strategy first we divide the dataset in k sub-sets and then holdout is applied on each of the subset as shown in Figure 6.1.

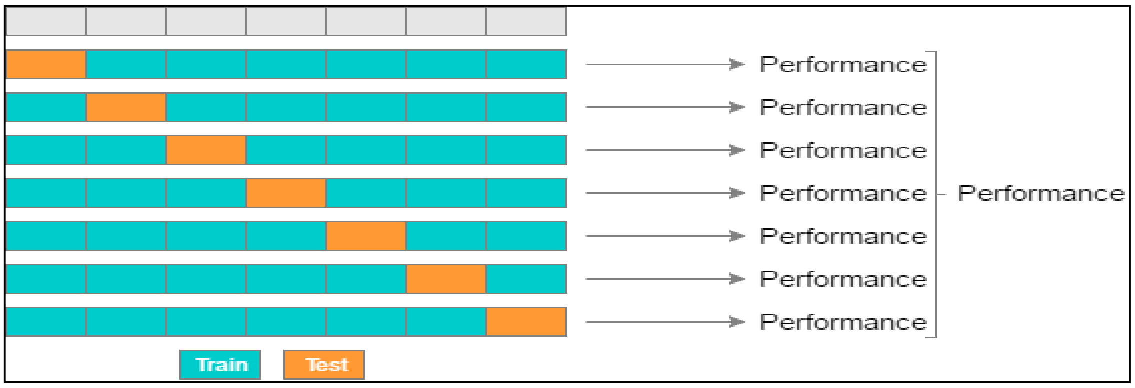


Figure 6.2K-fold with K=7 (Source:- ebc.cat/cross-validation-strategies)

In each k iterations a different subset is used for testing and rest all other are used for training. As in Figure 6.2 we can see that the complete dataset is divided into 7 (K=7) different sets and in every set, we are using different set of test set and other sets as training set in this process we are also calculating the performance of model for each step. Based on the performance we will get our best performance and that model would be our best model.

In K-means we won’t be using cross-validation we will be using elbow method to find the best number of K values, this will be discussed in detail in the next section.

Now moving to the algorithms that we have applied to our dataset starting with K-means and then we will discuss ALS and Naïve Bayes algorithm

6.3 K-means

There are three types of learning methods in machine learning: 1) Supervised Learning, 2) Unsupervised Learning and 3) Semi-Supervised Learning. Algorithms belonging to Unsupervised Learning technique have no variable to predict. K-means is one of the unsupervised learning techniques in this we create a cluster of data. Clustering is the task of grouping a set of items in a manner that objects in the same cluster is more similar to each other than to objects in another cluster. We can cluster almost anything and more similar the items are in the cluster the better our clusters are.

K-means clustering iteratively allocates every data point to its nearest cluster based on the features. In every iteration steps of the algorithm, each data point is assigned to its nearest cluster based on some distance, which is usually the *Euclidean distance*. The K-means algorithm attempts to minimize the objective function

$$\mathrm{TWSS} = \sum_{i=1}^{k} \sum_{x \in C_i} d(\mathbf{x}, \mathbf{m}_i)^2$$

This is total within cluster sum of squares. Here *d* refers to Euclidean distance and *m* is the centroid of the *ith* cluster.

In Cartesian coordinates, if $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$ are two points in Euclidean *n*-space then the Euclidean distance from *p* to *q* is given by

$$\begin{aligned} d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \end{aligned}$$

Source: Wikipedia on 27/07/2020

The basic steps that K-means clustering uses to arrange similar items into one cluster are as follows:

Step1: First we initialize the value of *K* which indicates that we want to divide data points into *k* clusters.

Step2: After initializing the value of *K* we partition objects into *k* clusters by assigning it to the cluster whose centroid is closest, using Euclidean distance (we can use different distance methods as well such as Squared Euclidean distance, Manhattan distance etc.).

Step3: Update the cluster centroids based on the distance we are using.

Step4: Repeat the Step2 to Step3 till either we reach the maximum number of iterations we defined or there is no change to the cluster centroid.

The above version of K-means algorithm is also known as Lloyd algorithm.

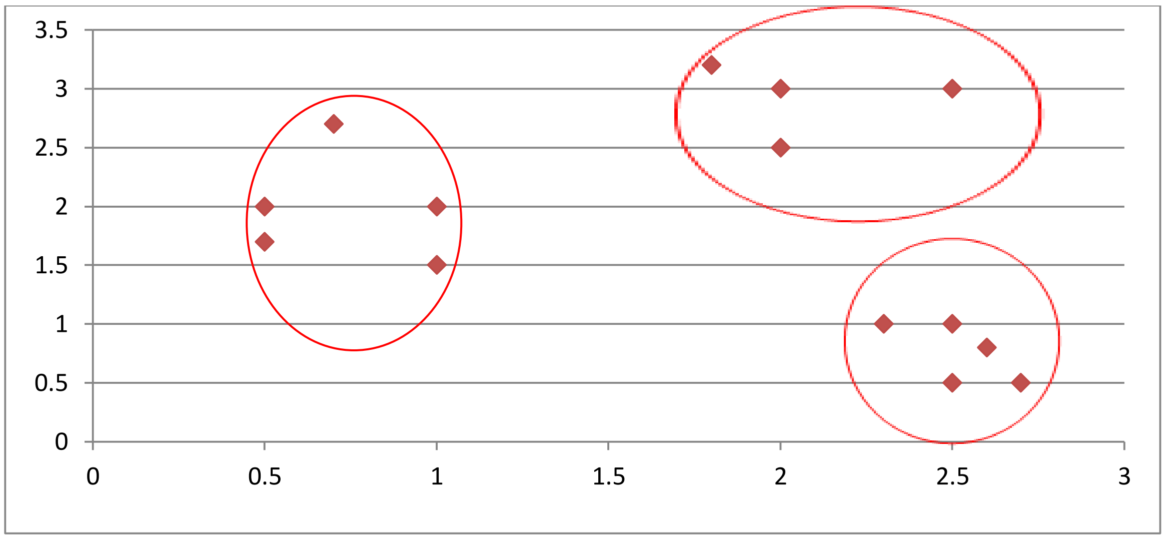


Figure 6.3Sample dataset for clustering (considering K=3)

In Figure 6.3 we can see that the data is clustered into 3 clusters based on the features. We can also see that the data in the same cluster are closer to each other than the data into the other clusters.

In our case we will be using *K-means* function provided by the package *pyspark.ml.clustering.*

```
from pyspark.ml.clustering import K-means

K-means= K-means(maxIter=5).setK(5).setSeed(1).setPredictionCol("cluster_prediction")
```

In the above code we are using maximum iterations as 5, value of k as 5 and prediction column as cluster_prediction. This will try to run K-means 5 times and set random value for centroid in first iterations, perform this step 5 times and set value of K as 5 and try to find the best cluster for our dataset. But before applying K-means clustering we have to do feature selection. We will be using product id (asin), reviewerID, total number of counts the reviewer has reviewed the item, Average rating given by the user for any item.

Feature selection in K-means is totally based on business logics and different people working on same dataset and using K-means might come up with different feature selection.

After feature selection we have to change strings to indexes as machine learning algorithm works only with numbers and doesn’t support string types and asin and revieweID are string in our dataset. We have to convert these strings into the integer type and for that we will be using string indexer one of the features provided by PySpark.

```
from pyspark.ml.feature import StringIndexer

revIDindexer = StringIndexer(inputCol="reviewerID", outputCol="reviewerID_index")

asinindexer = StringIndexer(inputCol="asin",outputCol="asin_index")
```

The above code will convert the strings into the indexes we will convert it back to the string once our computation is done.

After converting strings to indexes, we can go with the algorithm. But to get unbiased result we should perform one more step before going with our K-means algorithm. We should perform the scaling of the columns if we don’t perform scaling then it will be one feature centric, if any feature is having more weight than there will be no use of having multiple features in our model all the clusters will be created based on only that one feature. For example, we are calculating total number of reviews by reviewer (value could as big as 400 or 500) and considering this as a feature with ratings which will have value 1-5. Count will have more weight and clusters will be created based on the count only.

We will use MinMaxScaler to scale our data and after scaling we will use vector assembler to create a feature column to pass it to the K-means.

6.3.1 Evaluation of K-means

To find the best value of k for our clustering algorithm we will be using *Elbow Method.*

A fundamental step in unsupervised learning is to find the optimal number of clusters into which data may be clustered. The *Elbow method* is one of the most well-known methods to determine the optimal value of k. The idea behind elbow curve is to run K-means clustering for a range of clusters k and for each value we are calculating the sum of squared distances from each point to its assigned center. When these distances and value of k are plotted the plot looks like an arm then the “elbow” (the point of inflection of the curve) is the suggested value of k.

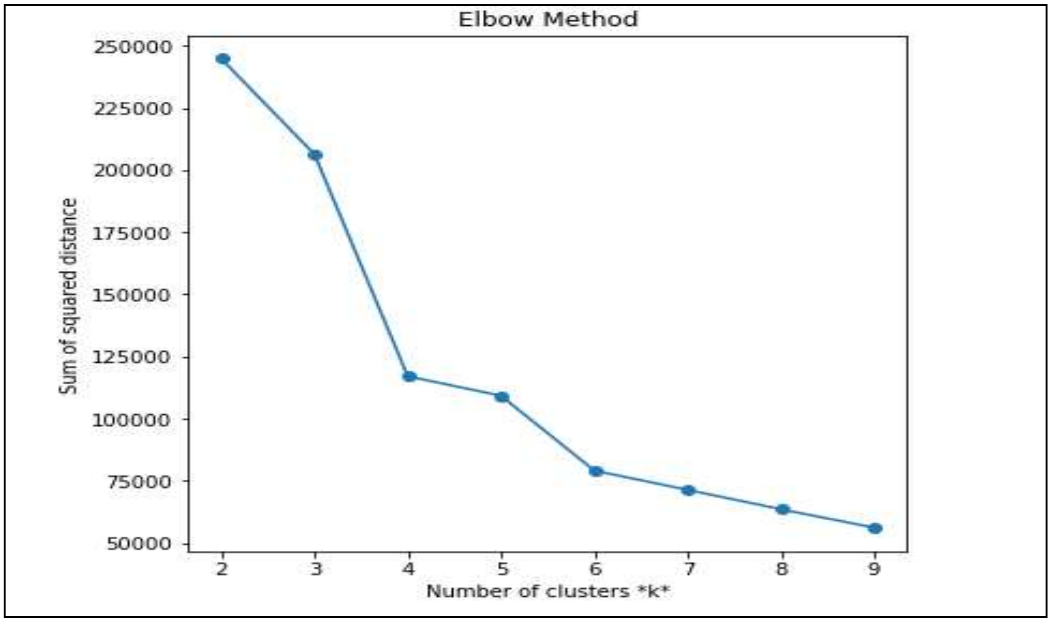


Figure 6.4 Elbow method to find value of k

From Fig 6.4 we can see that the inflection of the curve in the plot is at a point where value of k is in between 4-6. So, for clustering we will set k to 5. To plot the curve, we will use below code:

```
list_k = list(range(2,10))

for i in list_k:

   kmeans = KMeans(maxIter = 3).setK(i).setSeed(1)

   model_kmeans = kmeans.fit(new_df)

   list_k1.append(list_k)

   wcss.append(model_kmeans.computeCost(new_df))

import matplotlib.pyplot as plt

plt.figure(figsize=(6, 6))

plt.plot(list_k, wcss,marker = 'o')

plt.xlabel(r'Number of clusters *k*')

plt.ylabel('Sum of squared distance')
plt.title('Elbow Method')

plt.show()
```

We are using a list of values for k (2 – 10) and finding the compute cost for each value of k. After finding compute cost we are plotting it with values of k and creating an elbow curve. Our compute cost, for different values of is shown in Table 6.1:

Table 6.1 Table for Elbow curve

Value of k	Compute Cost
2	244800.1851454335
3	206331.70370572945
4	117044.72502699014
5	109216.62208377867
6	79092.58847594129
7	71366.14790194106
8	63547.530996008376
9	56165.15868072104

After finding the best k value we will create clusters based on the k value and divide the complete dataset.

In our clustering algorithm we will be using k = *5* and compute cost is: *109216.62208377867*. We will check the Silhouette distance to analysis how well a point fits into the cluster.

In silhouette measure we analyze how close the points in any cluster are to the points in the other clusters. When the silhouette coefficient is next to +1 that means that the points are far way to the other clusters and when it is next to 0 it means that the points are very close or intersecting other clusters. With value of K as *5* silhouettes distance for points in one cluster to points in the other cluster is 0.5307991752747083.

After creating clusters, we will recommend the items to the customer but before recommendation we will perform few more steps to predict whether the recommended item is good or bad for customer and for that we will use Naïve Bayes algorithm.

6.4 Naïve Bayes

Naïve Bayes is a simple multiclass classification algorithm with the assumptions of independence between every pair of features. Naïve Bayes can be trained very efficiently. Within a single pass to data, it computes the conditional probability distribution of each feature given label, and then it applies Bayes theorem to compute the conditional probability distribution of label given an observation and use it for prediction.

Naïve Bayes is called Naïve Bayes since it simplifies the calculations of the probabilities for each class to make their calculation tractable. Rather than trying to quantify each attribute's probabilities, they are presumed to be conditionally independent, provided the class. This is a very strong assumption that, in actual data, is most impossible, i.e. the attributes do not interact. The approach nonetheless performs surprisingly well in many applications in the real world.

Training a model using Naïve Bayes is very fast, since only the probabilities of each class with different input values need to be calculated.

Bayes theorem provides a way that we can calculate the probability of a set of data belonging to a given class given our prior knowledge. It is stated as

P (class | data) = P (data | class)*P (class) / P (data)

P (class | data) is the probability of the class given the provided data.

The basic steps that Naïve Bayes algorithm uses to predict highest probability for our outcome are as follows (when using single feature):

Step1: Calculate the prior probability for given class labels.

Step2: Find likelihood probability with each attribute for class.

Step3: Put these values in Bayes formula and calculate posterior probability.

Step4: See which class has a higher probability, given the input belongs to higher probability class.

In our work instead of calculating all the probabilities manually we will use package provided by *pyspark.ml.classification* as *NaiveBayes*.

```
from pyspark.ml.classification import NaiveBayes

df_final = df_final.withColumn(

            'label',when((col("overall").between(4, 5)),1).when((col("overall").between(0,3)),0))

nb = NaiveBayes(smoothing = 0.5)

vecAssembler_NB=VectorAssembler(inputCols=["asin_index", "reviewerID_index"], outputCol
= "features" , handleInvalid  =  "skip")

pipeline_NB = Pipeline(stages=[vecAssembler_NB, nb])

model = pipeline_NB.fit(training_df)

predictions = model.transform(test_df)
```

First, we create a label column and for that we will convert rating column into good or bad. We will mark rating between 4 and 5 as good and otherwise bad. After creating label, we will create feature column, as Naïve Bayes works with labeled points and features only than using pipeline, we will fit the Naïve Bayes algorithm to the training dataset to train the model.

6.4.1 Evaluation of Naïve Bayes

Before evaluation of Naïve Bayes, we will perform cross-validation to find out the best model with our training dataset. We will create parameter grid with different smoothing values for Naïve Bayes and train our model and try to find the best model with best smoothing parameters and good accuracy. We will be using package from *pyspark.ml.classification* as *NaïveBayes* and fit the model with our feature using pipeline as below:

```
from pyspark.ml.classification import NaiveBayes

nb = NaiveBayes(modelType="multinomial")
```

```
paramGrid = ParamGridBuilder().addGrid(nb.smoothing, [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]).build()

pipeline_NB = Pipeline(stages=[vecAssembler_NB, nb])
```

After cross-validation we will do evaluation with our best model and for that we will use *MulticlassClassificationEvaluator* from *pyspark.ml.classification*. To evaluate our model, we will use *'accuracy', 'Precision', 'Recall', 'F1 score', and 'confusion matrix'*.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

cvEvaluator=MulticlassClassificationEvaluator(labelCol="label", predictionCol= "prediction",
metricName="accuracy")
```

To calculate F1 score we will replace metricName in the evaluator from 'accuracy' to 'F1'. F1 score also known as balanced F-score or F-measure. The F1 score can be interpreted as a weighted average of the precision and recall, where F1-score reaches its best value at 1 and worst value at 0. The relative contribution of precision and recall to the F1 score are equal. The good F1 score means that we have low false positives and low false negatives. If a model is having F1 score as 1 then it is the best model whereas if it is 0 then it is a total failure.

We will check efficiency of our model with confusion matrix also. When it comes to building classification model, we should use confusion matrix, these are not only useful in model evaluation but also for model monitoring and model management. A confusion matrix is also known as error matrix, it is a summarization of table in a matrix form, used to access the performance of a classification model. The number of correct and incorrect predictions from our model in the test dataset are summarized into a matrix with count values and broken down by each class.

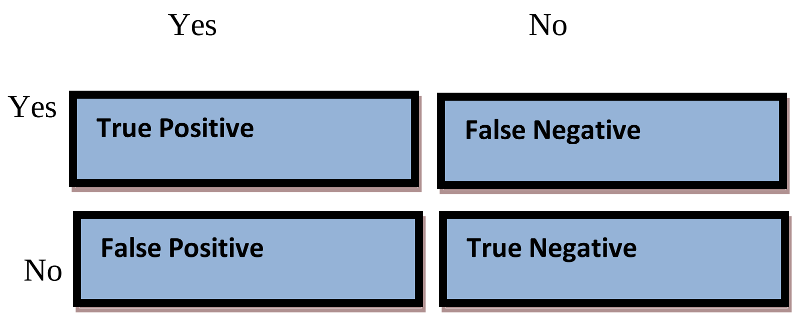


Figure 6.5 Structure of 2X2 confusion matrix

To plot confusion matrix we will be using multiclassmetrics of PySpark as below:

```
from pyspark.mllib.evaluation import MulticlassMetrics

prediction_evaluation = predictions.select("prediction", "label").rdd

metrics = MulticlassMetrics(prediction_evaluation)

print(metrics.confusionMatrix())
```

$$\begin{bmatrix} 244233 & 114362 \\ 50647 & 92531 \end{bmatrix}$$

From the confusion matrix we can calculate accuracy, precision, recall and F1 score to evaluate our model.

Precision: Precision is called as Positive Predicted values. The ratio of correct positive predictions to the total predicted values.

P = TP / TP + FP

Where TP is True positive and FP is False Positive.

P = 0.82824538795442

Recall: This is also called as sensitivity, Probability of Detection, True positive rate. The prediction of correct positive predictions to the total positive samples

R = TP / TP + FN

R = 0.68108311605014

Where FN is False Negative

Accuracy: Accuracy is defined as the ratio of correctly predicted sample from the complete dataset.

A = (TP + TN) / (TP + TN + FP + FN)

A = 0.67114810880617

Where TN is True Negative

F1 Score: It's a method of combining precision and recall into a single number. F1- Score is computed using a mean ("average") of precision and recall, but not the usual arithmetic mean. It uses the harmonic mean, which is given by formula [24]:

F1 = 2 x (Precision x Recall) / (Precision + Recall)

F1 = 0.7434

F1 score is always between precision and recall.

Table 6.2 Table for evaluation of Naive Bayes

Metric name	Value
Accuracy	0.67114810880617
Precision	0.82824538795442
F1 Score	0.7434
Recall	0.68108311605014

From the Table 6.2 we can see that accuracy of our model is close to 68% and F1 Score is also 0.7434**.**

We will also create a plot for different accuracy with different values of smoothing parameters that we have used in training our model using Naïve Bayes.

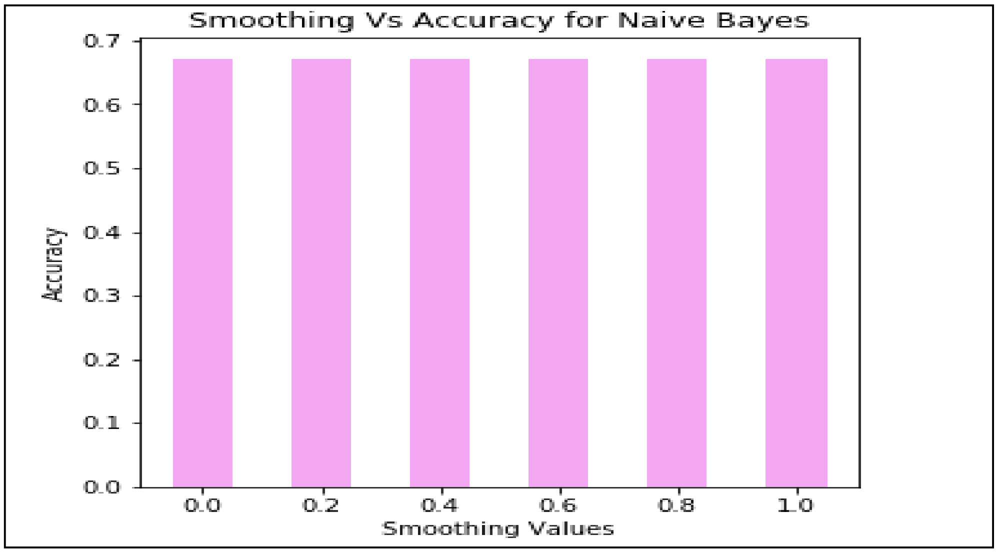


Figure 6.6 Smoothing Vs Accuracy for Naïve Bayes

From Figure 6.6 it shows that for our dataset there is no effect of smoothing values on accuracy.

6.5 Recommendation using K-means and Naïve Bayes

After creating clusters for all the items, we will recommend items to the user based on the cluster, reviewer belongs to. To recommend items reviewer has to enter their reviewer ID, then we will find which cluster that user belongs to, then create a list of top rating items from the cluster. After taking out all the items from the cluster we will check whether the items are predicted good for the customer or bad based on our classification algorithm (Naïve Bayes). Then based on three conditions we will recommend the items to user and those three conditions are as follows:-

Condition1: If the item is good for customer based on our prediction from Naïve Bayes then we will recommend the item to the user.

Condition2: If the item is predicted badly for the customer based on our prediction then we won’t recommend that particular item from the cluster and consider other items from the cluster.

Condition3: If the item is neither good nor bad, basically no prediction for the customer for particular items then we will recommend all the items from the cluster. The result would look like this:

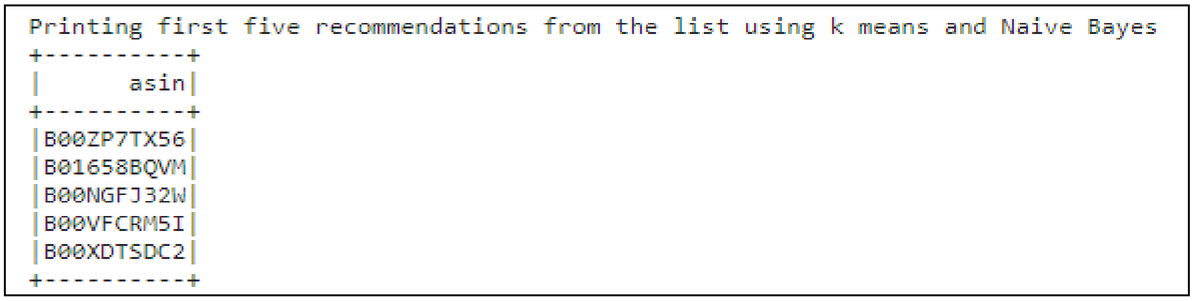
```
revID = input("Enter your reviewerID:")
```

We will provide revID as: A2V9BG2MDQVCYX

```
print("Printing first five recommendations from the list using k means and Naïve Bayes")

df3.show(5)
```

It will show the top 5 items for the user A2V9BG2MDQVCYX as shown in Figure 6.7



```
Printing first five recommendations from the list using k means and Naive Bayes
+----------+
|      asin|
+----------+
|B00ZP7TX56|
|B01658BQVM|
|B00NGFJ32W|
|B00VFCRM5I|
|B00XDTSDC2|
+----------+
```

Figure 6.7Recommendation using K-means

6.6 ALS (Alternate Least Square)

Many recommendation systems suggest items to user by utilizing the techniques of collaborative filtering based on historical records of items that the user have viewed, purchased or rated. Collaborative filtering techniques aim to fill in the missing entries of user-item matrix. *Spark-ml* currently supports model based collaborative filtering using Alternate Least Square (ALS).

Before starting with ALS first we should understand matrix factorization algorithm. Matrix factorization is simply a family of mathematical operations for matrices in linear algebra. Matrix Factorization is a factorization of matrix into product of matrices. In the case of collaborative filtering it works by decomposing the user-item interactions matrix into the product of two lower dimensionality rectangular matrices. One matrix is the user matrix where rows represent users and columns are latent factors. The other matrix is the item matrix where rows are latent factors and columns represent items.

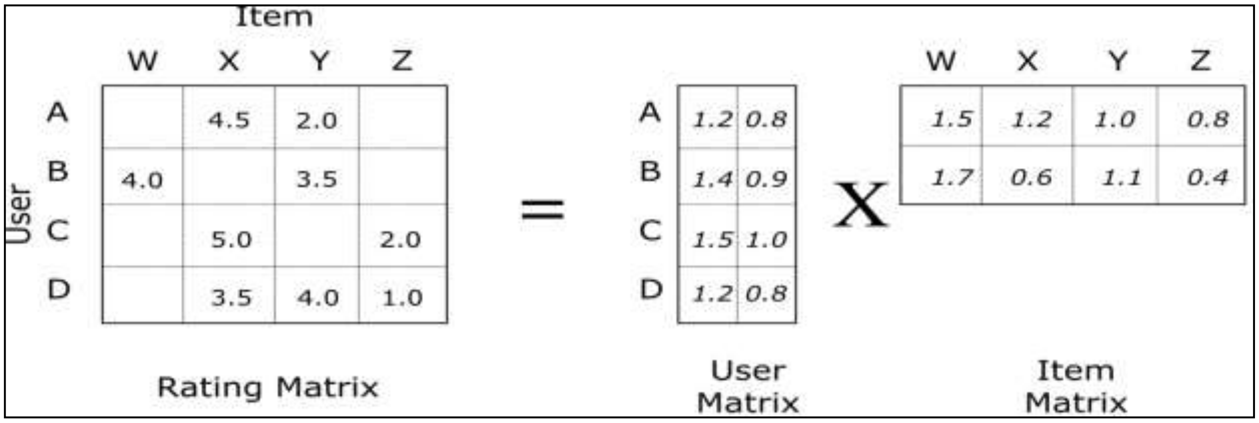


Figure 6.8Matrix Factorization (Source: towardsdatascience.com)

In Figure 6.8, the main goal is to predict the rating matrix as we can see that there are lot of missing values in that matrix, except for user *D* all the users have rated only two movies only out of 4 movies. To predict items our recommender system should know all the missing values then only it can recommend and to find the missing values it will perform the matrix multiplication of user matrix and item matrix.

$$\tilde{r}_{ui} = \sum_{f=0}^{nfactors} H_{u,f} W_{f,i}$$

Alternate Least Square is matrix factorization algorithm which runs in a parallel fashion. ALS is implemented in Apache Spark ML and built for large scale collaborative filtering problems.

The main objective of any matrix factorization algorithm is to minimize the error between true ratings and predicted ratings. ALS algorithm factorizes a given matrix *R* (rating matrix) into two factors *U* (user) and *V* (item) to minimize the error between true ratings and predicted ratings. In order to find the missing values in ratings matrix the below bellow problem is solved:

$$\arg\min_{U,V} \sum_{\{i,j|r_{i,j}\neq 0\}} \left(r_{i,j} - u_i^T v_j\right)^2 + \lambda \left(\sum_i n_{u_i} \|u_i\|^2 + \sum_j n_{v_j} \|v_j\|^2 \right)$$

Where λ being the regularization factor, n_{ui} being the number of items the user *i* has rated and n_{vj} being the number of items the item *j* has been rated. This regularization scheme to avoid overfitting is called weighted- λ- regularization.

With our recommender system we will be using ALS from *pyspark.ml.recommendation* to predict the missing ratings of items which reviewer has not reviewed and then based on that we will recommend the items to the user.

```
from pyspark.ml.recommendation import ALS

als_avg=ALS(userCol="reviewerID_index",itemCol="asin_index",ratingCol="Average",coldSt
artStrategy="drop",nonnegative=True).setPredictionCol("ALS_prediction_avg")
```

As in most of the machine learning algorithms, the Dataframe based API for ALS in spark only supports integers for user and item columns. So, as above again we have to convert these columns into their indexes and then we can use this in training our model. In this we are providing index of user ID column, index of item ID column and ratings given by the user with all these parameters we are also using another important parameter and that is *coldStartStrategy="drop"*.

When making predictions from the test data set or using the real-world data set it is very common that users or items in the data set was not present when we were training our model. When we try to predict items for the user or try to predict user for the items by default spark assigns NaN predictions and any NaN predicted value will result in NaN results for evaluation metric and this makes a model selection impossible. According to Apache Spark currently the only supported cold start strategies are NaN (the default behavior) and drop.

6.6.1 Evaluation of ALS

Before evaluating ALS algorithm, we will perform cross-validation to find out the best model with our dataset. For cross-validation we will be creating parameter grid for the cross-validation and use our model on this parameter grid to find the best values of our parameter so that we don't fall into overfitting of the model. To create a parameter grid we will use *ParamGridBuilder from pyspark.ml.tunning* package and for cross validation we will use *CrossValidator from pyspark.ml.tunning*.

```
from pyspark.ml.tuning import ParamGridBuilder

from pyspark.ml.tuning import CrossValidator

param_grid = ParamGridBuilder()\

    .addGrid(als_avg.rank, [100, 150])\

    .addGrid(als_avg.maxIter, [5])\

    .addGrid(als_avg.regParam, [ 0.09,0.01,0.5])\

    .build()

evaluator_ALS_avg=RegressionEvaluator(metricName="rmse",labelCol="Average",prediction
Col="ALS_prediction_avg")

cv = CrossValidator(estimator = als_avg,estimatorParamMaps = param_grid,evaluator =
evaluator_ALS_avg,numFolds = 3)
```

For cross validation we will be using 3 folds and try to find the best model from parameter grid and for evaluation we will be using *Root Mean Square Error* (RMSE). Based on our experimental analysis above best parameter for our model will be Rank = 150, MaxIter = 5 and RegParam as 0.01.

RMSE with this specification would be 2.15.

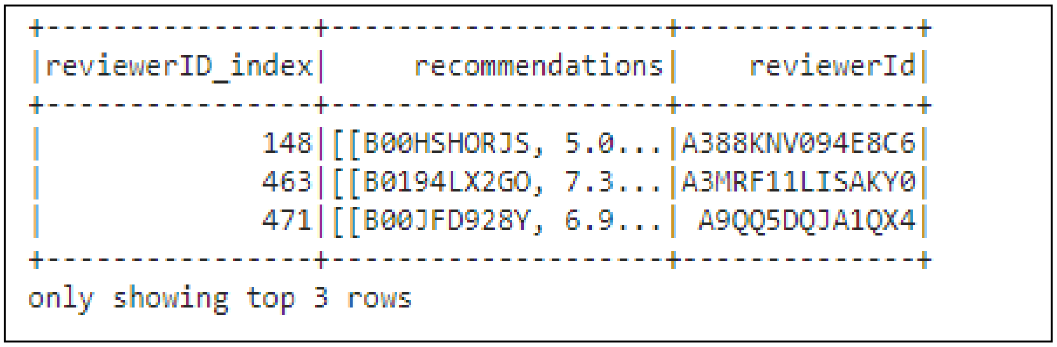
MAE (mean absolute error) is *1.6562256413032312*

6.6.2 Recommendation Using ALS

After getting best model from the cross validation we can use *recommendForAllUsers* function by ALS to recommend items for the entire set of users. We will pass 5 recommended items to the user in descending order.

```
user_recs_avg=model_ALS_avg.recommendForAllUsers(5)
```

Since we have used indexes of user and item so we have to convert indexes to the string back using *IndexToString* feature from *pyspark.ml.feature* and then we can show the items for user as shown in Figure 6.9.



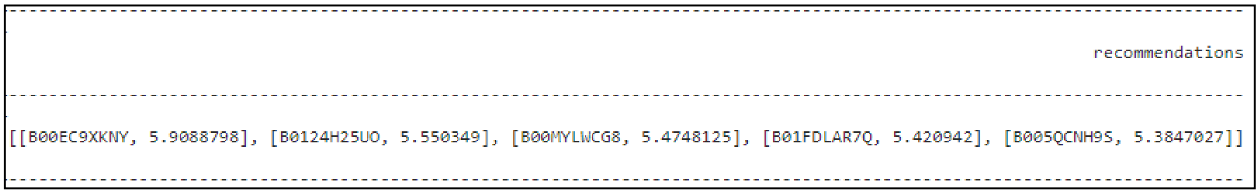
```
+----------------+--------------------+--------------+
|reviewerID_index|     recommendations|    reviewerId|
+----------------+--------------------+--------------+
|             148|[[B00HSHORJS, 5.0...|A388KNV094E8C6|
|             463|[[B0194LX2GO, 7.3...|A3MRF11LISAKY0|
|             471|[[B00JFD928Y, 6.9...| A9QQ5DQJA1QX4|
+----------------+--------------------+--------------+
only showing top 3 rows
```

Figure 6.9Recommendation Using ALS

In recommendations it will show items with their ratings for user in reviewerID there will be total 5 recommendations for the user.

For reviewerID:- A2V9BG2MDQVCYX the items would be:

final_recom_avg.select('recommendations').filter(final_recom_avg['reviewerID']=='A2V9BG2MDQVCYX').show(2,300)



```
------------------------------------------------------------------------------------------------------------

                                                                                             recommendations

------------------------------------------------------------------------------------------------------------

[[B00EC9XKNY, 5.9088798], [B0124H25UO, 5.550349], [B00MYLWCG8, 5.4748125], [B01FDLAR7Q, 5.420942], [B005QCNH9S, 5.3847027]]

------------------------------------------------------------------------------------------------------------
```

Figure 6.10 Recommendation for A2V9BG2MDQVCYX using ALS

In Figure 6.10 we can see 5 items recommended for the reviewer *'A2V9BG2MDQVCYX'* all having rating as 5.

Chapter 7 Conclusion and Future Work

7.1 Conclusion

It is very rare that customer has first decided what to buy and then search for it online, a customer is sure about their needs and slightly know about what product can satisfy them. There is a very fine line between trying to fool the customer and cross-selling and here better recommendation system plays a very crucial role.

The motivation behind this research was to find the best method for recommendation of items to customer based on their shared features for e-commerce companies. We combined two learning techniques of machine learning i.e. supervised learning and unsupervised learning for our recommendation system. We used K-means for clustering the reviewer with similar features and classification algorithm (Naïve Bayes) to predict whether the predicted item from the clustering is good for the customer or not. In this method we first created the clusters and then based on clusters we recommend the items using Naïve Bayes algorithm for predicting the item to be good or bad based on past experience of the model trained.

For recommendations we also used ALS method, which is very common for recommendation, it uses matrix factorization method to predict the missing ratings and then recommend the best item for the user. In ALS we are creating three matrices i.e. item matrix, user matrix, and rating matrix. Using item matrix, user matrix and some smoothing parameters we are predicting the rating matrix. Most of the recommendation system in some big companies like Netflix and Flipkart uses ALS method for recommendation obviously with better features and better smoothing parameters. There are two methods of recommendation ALS and SGD (Stochastic Gradient Descent) which are widely used. For Explicit feedback such as Ratings given by user at IMDB or Netflix, ALS is used and for Implicit feedback such as purchase history, browsing patterns SGD is used.

7.2 Future Work

There is always a scope of improvements; nothing is perfect so as is the case with machine learning algorithms. There can be lot of future work that can be done with the platform that we set in this thesis; some of them are noted below:

1) First and very important thing to work with Big Dataset is we should have a spark into our system and if we are using spark on cloud provided by Google or Amazon that would be very helpful. It will reduce the task of configuring spark while running spark submit to run our application, we just have to run our code and cloud will handle all the configuration part. We can also monitor how our distributed system is working on web UI.
2) As in section 6.4.2 for ALS we saw that RMSE is bit high we can use more values in our parameter grid to tune our model.
3) Now coming to the data set if we have the purchase history of customer, we can use that as well for recommending items and creating our cluster of reviewers for items to be recommended with K-means algorithm.
4) If we have few more columns in our dataset, for user, such as gender, age and location then we can use these columns as well to improve clustering accuracy.
5) We can also use the sentiments from the review text or summary text and create our own score for that item and can use that score for recommending, as sometimes you don’t even want to give rating as 1 but you don’t have any option, since 1 is minimum rating that you can give, so in this case our sentiment score can help.
6) Instead of K-means we can also explore other methods such as KNN algorithm to find if the two customers are similar or not.
7) There are two methods of recommendation which are widely used one is ALS which we have used and second is Stochastic Gradient Descent SGD method we can also explore this method for recommendation.
8) As we saw in Chapter 4 that in the dataset there are around 64% of reviews were given 5 star, which makes it unbalanced dataset, to balance the dataset we can consider vote column, and see if any particular review is 5 star and also voted high by other reviewer,

then we will consider that as 5 star product and based on that we can do our recommendation.

9) As we have used Amazon_Fashion and All_Beauty from Amazon product review, we can also use more files and configure spark accordingly to work with more files for cross-recommendation following the same steps as discussed in the thesis.
10) In our research we are using Naïve Bayes algorithm with K-means for all the clusters, but we can also use some other algorithm based on the cluster such as for cluster 1 we can use Naïve Bayes and for cluster 2 we may use logistic or random forest algorithm based on the distribution of items in that particular cluster.

References

[1] Pavels Goncarovs, “Data Analytics in CRM Process: A Literature Review”, *Riga Technical University,* vol. 20, pp. 103–108, Dec 2017

[2] Bernard F.Kubiak, “Cross-And Up-Selling Techniques in E-Commerce Activities”, *University of Gdansk,* vol. 15, no. 3, Dec 2010

[3] Ren-Qian Zhang, Qi-Qi Wang, Yi-Ye Zhang, Hai-Tao Zheng and Jie Hu, “Estimation of Jointly Normally Distributed Demand for Cross-Selling Items in Inventory Systems with Lost Sales”, *School of Economics and Management, Beihang University,* Jul 2019.

[4] Wagner A.Kamakura, *“Cross-Selling”, Journal of Relationship Marketing,* 6:3-4, 41-58, 2008

[5] R. A. Fisher, “Properties and applications of Hh functions”, *Introduction to Mathematical Tables,* vol. 1, pp. xxiv–xxxv, 1931.

[6] A. C. Cohen, “On Estimating the Mean and Standard Deviation of Truncated Normal Distributions”, *Journal of the American Statistical Association*, vol. 44, no. 248, pp. 518–525, 1949.

[7] A. K. Gupta, “Estimation of the mean and standard deviation of a normal population from a censored sample”, *Biometrika*, vol. 39, no. 3-4, pp. 260–273, 1952.

[8] S. Nahmias, “Demand estimation in lost sales inventory systems”, *Naval Research Logistics (NRL)*, vol. 41, no. 6, pp. 739–757, 1994.

[9] Gemius (Jul,2020), E-Commerce w Polsce 2019 [Online] .Available: http://gemius.pl.

[10] Kubiak B. F.*, “ Strategia informatyzacji współczesnej organizacji. Teoria i Praktyka.*

Wydawnictwo Uniwersytetu Gdańskiego, Gdańsk 2003.

[11] Meier A., Werro N., Albrecht M., Sarakinos M., “Using a Fuzzy Classification Query Language for Customer Relationship Management”, Proceedings of the 31st international

conference on Very large data bases, ACM, Trondheim, 2005.

[12] Netflix (2020, Jul). Netflix Prize [Online]. Available: https://www.netflixprize.com/

[13] Apache Spark (2020, Aug 4).Ml Collaborative Filtering [Online]. Available: https://spark.apache.org/docs/2.3.0/ml-collaborative-filtering.html

[14] Imad Dabbura (2020, Jul 20). K-means Clustering: Algorithm, Applications, Evaluation Methods and Drawbacks. Available: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

[15] Anon. (2020, Jun 25). How To Install Spark on Ubuntu [Online]. Available: https://phoenixnap.com/kb/install-spark-on-ubuntu

[16] E. Berk,U. G¨urler, and R. A. Levine, “Bayesian demand updating in the lost sales newsvendor problem: a two-moment approximation”, *European Journal of Operational Research*, vol. 182, no. 1, pp. 256–281, 2007.

[17] A. Jain, N. Rudi, and T. Wang, “Demand estimation and ordering under censoring: Stock-out timing is (almost) all you need”, *Operations Research*, vol. 63, no. 1, pp. 134–150, 2015.

[18] Anon. (2020, Jul 24). Wells Fargo Account Fraud Scandal [Online]. Available: https://en.wikipedia.org/wiki/Wells_Fargo_account_fraud_scandal

[19] Dina Gerdeman(2020, Jul 24). Better By The Bundle? [Online]. Available: https://hbswk.hbs.edu/item/better-by-the-bundle

[20] Charles (2020, Jul 25). Amazon Can Teach Cross-Selling [Online]. Available: https://predictableprofits.com/amazon-can-teach-cross-selling/

[21] Reggie Van Lee, Lisa Fabish, Nancy McGaw (2020, Aug 4). The Value of Corporate Values [Online]. Available: https://www.strategy-business.com/article/05206?gko=987a9

[22] BBC News (2020, Aug). Amazon at 25: The Story of A Giant. [Online]. Available: https://www.bbc.com/news/business-48884596#:~:text=1995%3A%20Amazon%20launches%20with%20online%20book%20

sales&text='%20And%20that%20was%20true%2C%22,bricks%2Dand%2Dmortar%20rivals.

[23] R.He, J. McAuley, “Ups and downs: Modeling the visual evolution of fashion trends with one class collaborative filtering”,*www,* 2016

[24] Boaz Shmueli (2020, Aug 4). Multi-Class Metrics Made Simple, PartII: the F1- Score [Online]. Available: https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1

[25] Rothfeder, Jeffrey, “Trend: Cross-selling”, *Ziff Davis CIO Insight,* Jul 2003

[26] George Box, “Science and Statistics”,1976

Appendix

Appendix A:- Python Code

Note:-

1) All the files are placed at ICHEC server at navigation:- /ichec/work/mucom001c/Amazon/review/
2) The Spark session for the python code is set to work with ShmemQ node of ICHEC which is having 1.5 TB of RAM and 40 cores.
3) If running the code using slurm script (Script in Appendix B) then the spark needs to be configured as discussed in chapter 5 in the directory running the code.
4) Some Common errors that might come if spark is not configured properly or using node other then ShmemQ, are: *Java Server Connection Lost, Java Heap Memory Issue, Executor Heartbeat lost if executor is idle for very long, Network Timeout and Spark Driver memory is low.*

```
import os
def setup_my_environment():
    import os
def setenv(var, val):
    os.environ[var] = val
def prepend_path(var, val):
    old_val = os.environ.get(var, '')
    os.environ[var] = val + ":" + old_val
def setup_java():
    PKG_ROOT='/ichec/packages/java/8'
    setenv('JAVA_PATH', PKG_ROOT)
    setenv('JAVA_HOME', PKG_ROOT)
    prepend_path('PATH', PKG_ROOT + '/bin')
    prepend_path('MANPATH', PKG_ROOT + '/man')
    prepend_path('CPATH', PKG_ROOT + '/include')
def setup_spark():
    PKG_ROOT='/ichec/packages/spark/2.3.3/kay/spark-2.3.3'
    setenv('SPARK_DIST_CLASSPATH', PKG_ROOT + 'spark-2.3.3-bin-kay-spark')
    prepend_path('PATH', PKG_ROOT + PKG_ROOT + 'spark-2.3.3-bin-kay-spark/bin')
setup_java()
setup_spark()
setup_my_environment()
from pyspark import SparkConf
from pyspark.context import SparkContext
from pyspark.sql import SQLContext
from pyspark.sql import SparkSession
spark = SparkSession \
    .builder \
    .config("spark.executor.memory", "100g") \
    .config("spark.executor.cores","3")\
    .config("spark.driver.memory", "100g") \
    .config("spark.driver.maxResultSize","100g")\
    .config("spark.executor.instance","12")\
    .config('spark.sql.shuffle.partitions',"128")\
    .config("spark.sql.crossJoin.enabled","true")\
    .config("spark.debug.maxToStringsFields","100")\
    .appName("AmazonCrossSelling") \

    .getOrCreate()

 df1 = spark.read.json("/ichec/work/mucom001c/Amazon/review/All_Beauty.json")

df2 = spark.read.json("/ichec/work/mucom001c/Amazon/review/AMAZON_FASHION.json")

df=df1.join(df2,["asin","image","overall","reviewText","reviewTime","reviewerID","reviewerNa
me","summary","unixReviewTime","verified","vote"],"full_outer").drop("style")

df.dtypes

# To check null values in the dataset and convert the Boolean value of verified to 1 and 0

import pyspark.sql.functions as F

df.select([F.count(F.when(F.col(c).isNull(), c)).alias(c) for c in df.columns]).show()

df = df.fillna({'vote':'1'})

df = df.withColumn('verified', F.when(df.verified == 'false', 0).otherwise(1))

df.select([F.count(F.when(F.col(c).isNull(), c)).alias(c) for c in df.columns]).show()

df_new = df.select('asin','overall','reviewerID','verified','vote')

feature_group = ["reviewerID","asin","verified"]

df_avg =df.groupby(feature_group).agg(F.mean("overall").alias("Average"))

df_count = df.groupby(feature_group).count()

df_final = df_avg.join(df_count,feature_group)

df_vote = df.groupby(feature_group).agg(F.sum("vote").alias("Total_Votes"))

df_final = df_final.join(df_vote,feature_group)

df_final = df_final.join(df_new,feature_group)

df_final.dtypes

# To convert reviewerID and asin from strings to indexes so that to use in K-means

from pyspark.ml.feature import StringIndexer

from pyspark.ml import Pipeline
```

```
revIDindexer = StringIndexer(inputCol="reviewerID", outputCol="reviewerID_index")

asinindexer = StringIndexer(inputCol="asin",outputCol="asin_index")

pipeline = Pipeline(stages=[asinindexer,revIDindexer])

df_final = pipeline.fit(df_final).transform(df_final)

# To Scale the columns

from pyspark.ml.feature import MinMaxScaler

from pyspark.ml.feature import VectorAssembler

from pyspark.ml import Pipeline

from pyspark.sql.functions import udf

from pyspark.sql.types import DoubleType

unlist = udf(lambda x: round(float(list(x)[0]),3), DoubleType())

for i in ["reviewerID_index","asin_index","count","Average","Total_Votes"]:

    assembler = VectorAssembler(inputCols=[i],outputCol=i+"_Vect",handleInvalid = "skip")

    scaler = MinMaxScaler(inputCol=i+"_Vect", outputCol=i+"_Scaled")

    pipeline = Pipeline(stages=[assembler, scaler])

    df_final=pipeline.fit(df_final).transform(df_final).withColumn(i+"_Scaled",
unlist(i+"_Scaled")).drop(i+"_Vect")

# Creating features for K-means

from pyspark.ml.clustering import KMeans

from pyspark.ml.feature import VectorAssembler

from pyspark.ml.evaluation import ClusteringEvaluator

vecAssembler=VectorAssembler(inputCols=["reviewerID_index_Scaled","asin_index_Scaled","
count_Scaled","Average_Scaled","Total_Votes_Scaled"], outputCol="features",handleInvalid =
"skip")

new_df = vecAssembler.transform(df_final)

new_df=new_df.drop('count','Total_Votes','reviewerID_index','asin_index')
```

```
# Plotting Elbow Curve
wcss = []
list_k1 = []
list_k = list(range(2,10))
for i in list_k:
    kmeans = KMeans(maxIter = 3).setK(i).setSeed(1)
    model_kmeans = kmeans.fit(new_df)
    list_k1.append(list_k)
    wcss.append(model_kmeans.computeCost(new_df))
    print("Within Set Sum of Squared Errors = " + str(wcss))
    print("Value of k = " + str(i))
import matplotlib.pyplot as plt
plt.figure(figsize=(6, 6))
plt.plot(list_k, wcss,marker = 'o')
plt.xlabel(r'Number of clusters *k*')
plt.ylabel('Sum of squared distance')
plt.title('Elbow Method')
plt.show()
#kmeans algorithm
kmeans = KMeans(maxIter = 5).setK(5).setSeed(1).setPredictionCol("cluster_prediction")
model_kmeans = kmeans.fit(new_df)
# calculating sum of squared error and Silhouette distance
wssse = model_kmeans.computeCost(new_df)
print("Within Set Sum of Squared Errors = " + str(wssse))
evaluator_kmeans = ClusteringEvaluator().setPredictionCol("cluster_prediction")
```

```
df_predictions = model_kmeans.transform(new_df)

silhouette_kmeans = evaluator_kmeans.evaluate(df_predictions)

print("Silhouette with squared euclidean distance = " + str(silhouette_kmeans))

revID = input("Enter your reviewerID:")

cluster=df_predictions.filter(df_predictions['reviewerID']==revID).select('cluster_prediction').d
istinct()

df_predictions.createOrReplaceTempView("KmeansAmazonDataset")

df3 = spark.sql("SELECT asin from KmeansAmazonDataset where Average >= 4 and
cluster_prediction = '{0}'".format(cluster.first().cluster_prediction))

#starting Naive Bayes

from pyspark.sql.functions import *

from pyspark.sql.functions import when

df_final = df_final.withColumn(

    'label',

     when((col("overall").between(4, 5)),1).when((col("overall").between(0,3)),0)

)

(training_df,test_df)=df_final.randomSplit([0.6, 0.4])

#Naive Bayes algorithm for recommendation of products cross validation with various
smoothing values.

from pyspark.ml.classification import NaiveBayes

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

vecAssembler_NB = VectorAssembler(inputCols=["asin_index", "reviewerID_index"],
outputCol="features")

nb = NaiveBayes(modelType="multinomial")

paramGrid = ParamGridBuilder().addGrid(nb.smoothing, [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]).build()

pipeline_NB = Pipeline(stages=[vecAssembler_NB, nb])

(training_df,test_df)=df_final.randomSplit([0.6, 0.4])
```

```
cvEvaluator = MulticlassClassificationEvaluator(labelCol = "label", predictionCol=
"prediction", metricName="accuracy")

cv = CrossValidator(estimator=pipeline_NB, estimatorParamMaps=paramGrid,
evaluator=cvEvaluator,numFolds = 4)

NB_Model = cv.fit(training_df)

NB_Predictions = NB_Model.transform(test_df)

#Naive Bayes without Cross-Validation

from pyspark.ml.classification import NaiveBayes

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

from pyspark.sql.functions import col

nb = NaiveBayes(smoothing = 0.5)

vecAssembler_NB=VectorAssembler(inputCols=["asin_index","reviewerID_index"],
outputCol="features",handleInvalid = "skip")

pipeline_NB = Pipeline(stages=[vecAssembler_NB, nb])

model = pipeline_NB.fit(training_df)

predictions = model.transform(test_df)

evaluator=MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction",metricName="accuracy")

evaluator.evaluate(predictions)

df4=predictions.filter(predictions['reviewerID']==revID).select(col("reviewerID"),col("asin").a
lias("item"),col("label"),col("overall"),col("prediction"))

print("Printing first five recommendations from the list using k means and Naive Bayes")

df3.select('asin').distinct().show(5)

# Now applying ALS method for recommendation

# recommendation using ALS method

df_ALS = new_df.select(new_df['asin'],new_df['reviewerID'],new_df['Average'])

from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS

from pyspark.ml.feature import StringIndexer

from pyspark.ml import Pipeline

from pyspark.sql.functions import col

revIDindexer = StringIndexer(inputCol="reviewerID", outputCol="reviewerID_index")

asinindexer = StringIndexer(inputCol="asin",outputCol="asin_index")

pipeline = Pipeline(stages={asinindexer,revIDindexer})

transformed = pipeline.fit(df_ALS).transform(df_ALS)

(training_df,test_df)=transformed.randomSplit([0.6, 0.4])

#finding best model for ALS method using average

from pyspark.ml.tuning import ParamGridBuilder

from pyspark.ml.tuning import CrossValidator


als_avg=ALS(userCol="reviewerID_index",itemCol="asin_index",ratingCol="Average",coldSt
artStrategy="drop",nonnegative=True).setPredictionCol("ALS_prediction_avg")

param_grid = ParamGridBuilder()\

    .addGrid(als_avg.rank, [100,150])\

    .addGrid(als_avg.maxIter, [5])\

    .addGrid(als_avg.regParam, [0.09,0.01,0.5])\

    .build()

evaluator_ALS_avg=RegressionEvaluator(metricName="rmse",labelCol="Average",prediction
Col="ALS_prediction_avg")

cv = CrossValidator(estimator = als_avg,estimatorParamMaps = param_grid,evaluator =
evaluator_ALS_avg,numFolds = 3)

ALS_model_CV_avg = cv.fit(training_df)

ALS_model_avg = ALS_model_CV_avg.bestModel
```

```
ALS_predictions_avg = ALS_model_avg.transform(test_df)

rmse_ALS_avg=evaluator_ALS_avg.evaluate(ALS_predictions_avg)

print("RMSE considering Average in ALS="+str(rmse_ALS_avg))

# Print evaluation metrics and model parameters

print ("**Best Model**")

print ("RMSE = ", rmse_ALS_avg)

print (" Rank: ", ALS_model_avg.rank)

print (" MaxIter: ", ALS_model_avg._java_obj.parent().getMaxIter())

print (" RegParam: ", ALS_model_avg._java_obj.parent().getRegParam())

# Recommending 5 items for every user

user_recs_avg=ALS_model_avg.recommendForAllUsers(5)

from pyspark.ml.feature import IndexToString

from pyspark.ml.feature import StringIndexerModel

user_labels = revIDindexer.fit(df_ALS).labels

product_labels = asinindexer.fit(df_ALS).labels

user_id_to_label = IndexToString(inputCol="reviewerID_index", outputCol="reviewerId",
labels=user_labels)

n = 5

product_labels_ =F.array(*[F.lit(x) for x in product_labels])

recommendations=F.array(*[F.struct(product_labels_[F.col("recommendations")[i]["asin_inde
x"]].alias("asin"),F.col("recommendations")[i]["rating"].alias("rating")) for i in range(n)])

final_recom_avg = user_id_to_label.transform(user_recs_avg)

final_recom_avg = final_recom_avg.withColumn("recommendations",recommendations)

final_recom_avg.show(3)

final_recom_avg.select('recommendations').filter(final_recom_avg['reviewerID']==revID).show
(2,False)
```

Appendix B:- Slurm Script

```
#!/bin/sh


#SBATCH --time=01:00:00

#SBATCH --nodes=1

#SBATCH -A mucom001c

#SBATCH -p ShmemQ


module load conda/2

source activate mynewenv2

module load java

module load spark

module load intel

module load taskfarm


export SPARK_HOME='/ichec/home/users/alokkumarsin/myfiles/spark-2.3.3-bin-kay-spark'

export JAVA_HOME='/ichec/home/users/alokkumarsin/myfiles/8'


/ichec/home/users/alokkumarsin/myfiles/spark-2.3.3-bin-kay-spark/sbin/start-master.sh

/ichec/home/users/alokkumarsin/myfiles/spark-2.3.3-bin-kay-spark/sbin/start-slaves.sh

export SPARK_WORKER_CORES=3

export SPARK_EXECUTOR_CORES=3

export SPARK_MASTER_WEBUI_PORT=8081

export SPARK_WORKER_INSTANCES=12
export SPARK_EXECUTOR_INSTANCES=12

export SPARK_MASTER_MEMORY=1024M

export SPARK_EXECUTOR_MEMORY=100G

export SPARK_WORKER_MEMORY=15G


spark-submit  amazon_cross_yarntesting.py

/ichec/home/users/alokkumarsin/myfiles/spark-2.3.3-bin-kay-spark/sbin/stop-slaves.sh

/ichec/home/users/alokkumarsin/myfiles/spark-2.3.3-bin-kay-spark/sbin/stop-master.sh
```