AIN SHAMS UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

CSE412: Selected Topics in Computer Engineering

Apriori Algorithm

Submitted by:

Team 26:
Abdelrahman Mamdouh Khalil Al-Wali
Abdullah Ahmed Fathy Hamed
Abdullah Khairy Ismail
Ahmed Khalid Salah Eldin
Abdullah Emad El-din Mohammed Hajjaj

Supervised by:

Dr. Gamal A. Ebrahim

2020

Abstract

Nowadays Data mining has a lot of e-Commerce applications. The key problem is how to find useful hidden patterns for better business applications in the retail sector. For the solution of these problems, The Apriori algorithm is one of the most popular data mining approach for finding frequent item sets from a transaction dataset and derive association rules. Rules are the discovered knowledge from the database. Finding frequent item set (item sets with frequency larger than or equal to a user specified minimum support) is not trivial because of its combinatorial explosion. Once frequent item sets are obtained, it is straightforward to generate association rules with confidence larger than or equal to a user specified minimum confidence. The paper illustrating apriori algorithm on simulated database and finds the association rules on different confidence values.

Table of Contents

Table of Figures

1. Introduction

The Apriori algorithm is one of the most influential apriori for mining association rules. The basic idea of the Apriori algorithm is to identify all the frequent sets. Through the frequent sets, derived association rules, these rules must satisfy minimum support threshold and minimum confidence threshold.

1.1 Purpose

The purpose of this project is to describe in details the importance of Apriori algorithm along with technical implementation of the algorithm itself.

1.2 List of definition

- Dataset:
  It is a collection of data records where each record represents a transaction. It consists of a Primary Key field along with several other items related to the transaction.

- Itemset:
  Each transaction consists of a set of items known as itemset.

- Frequent Itemset:
  It is a specific itemset whose support is greater than some user-specified minimum support.

- Support:
  This says how popular an itemset is, as measured by the proportion of transactions in which an itemset appears.

- Minimum Support:
  It is a parameter supplied to the Apriori algorithm in order to prune candidate rules by specifying a minimum lower bound for the Support measure of resulting association rules.

- Confidence:
  This explains how likely a certain item is purchased when another item is purchased. This defines association between the items.

- Lift:
  The lift value is a measure of importance of a rule. The lift value of an association rule is the ratio of the confidence of the rule and the expected confidence of the rule.

- Leverage:
  Leverage measures the difference of the items appearing together in the dataset and what would be expected if the items where statistically dependent. The rational in a sales setting is to find out how many more units are sold than expected from the independent sells.

1.3 Overview

The report is divided into four main sections, the first section focuses on a general overview of the project idea and a summarized introduction (Abstract and Introduction), the second section focuses on the project goals, intended benefits and it's beneficiaries; highlighting the intended goals of the project along with a detailed project description (Beneficiaries, Project Aims and Objectives), the third section focuses on the technical details of the project including the Project Phases (Detailed Project Description, Development Environment and System Architecture), the fourth section highlights the results (test cases used, and results out of them and the conclusion).

2. Beneficiaries

The social media and technology companies such as Facebook, Twitter, Amazon, etc. are the main beneficiaries of this algorithm and other similar algorithms. They depend on those algorithms to deal and cope with Big Data issues. As mentioned before, Apriori algorithm is a classical algorithm in data mining. It is used for mining frequent itemsets and relevant association rules. It is devised to operate on a database containing a lot of transactions.

3. Project Aims and Objectives

The aim of the project is to find relations between data frames in a given dataset which can be used to extract useful information from transactions; extracting associations between the data and sorting them according to the strength of those associations.

The extracted information can be directed towards useful businesses related to social media and technology companies such as Facebook, Twitter, Amazon, etc.

The main objective of this project is to:
1) Process this dataset [http://www.liacs.nl/~putten/library/cc2000/].
2) Extract useful relations and rules using Apriori algorithm.
3) Export the extracted rules as human readable format to a PDF file, text file or in the terminal output.

4. Detailed Project Description

Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for Boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Apriori uses breadth-first search and a Hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length **K** from item sets of length **K-1**. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent **K**-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates.

In project implementation, we divide code over 3 files:

- **1. Config.py**

  It consists of three sections:

  - **Data Config:** To select data directory and name of data file, number of your first attribute as it in the data file and name of the output csv file containing our attributes columns only.
  - **Sorting of rules:** To select sorting according to Lift or Leverage.
  - **HTML style for rules:** Css style code for styling output HTML file

- **2. Utils.py**

  It consists of definition of all used functions in the project:

  - **Data_extract:** To extract specific attributes from a data file based on configuration done in config.py and return dataframe containing the attributes.

  - **Support_lvl:** Get support of each level by giving the dataframe and return list of dictionaries that contain support of each sub attribute.

  - **Elimination_of_values:** Elimination of values less than minimum support from list of dictionaries and return filtered list of dictionaries.

- **Elimination_of_columns:** Elimination of empty columns in dataframe after elimination of values that less than minimum support as may some attributes that have no values exceeds the minimum support.

- **Make_combination:** To make combinations between attributes to higher the level as it takes current level itemsets (K) and return next level (K+1).

- **Collective_fun:** It takes Make_combination, Support_lvl, Elimination_of_values, Elimination_of_columns and returns lvl dataframe with only columns that exceeds minimum support.

- **Return_last_lvl_values:** Return only last lvl all values with support greater than minimum support.

- **Return_support:** Return support of specific combinations.

- **Confidence_of_rule:** Calculate confidence of rule.

- **Lift_of_rule:** Calculate lift of rule.

- **Leverage_of_rule:** Calculate leverage of rule.

- **Make_rules_and_prioritize:** Return all rules that have more than minimum confidence and prioritize them by lift or leverage descendingly.

- **DataFrame_to_HTML:** Converting output rules in dataframe into an HTML.

➢ 3. Main.py

To take User inputs and combine functions together:

- **User inputs:** Take minimum support in fraction (as 0.4) or number of appearance (as 322) and minimum confidence in fraction (as 0.6) or percentages (as 70%).

- Then calling four functions (data_extract, return_last_lvl_values, make_rules_and_prioritize, DataFrame_to_HTML) respectively.

5. Project Phases

Agile Methodology was chosen because it's not iterative, so the developers can work concurrently and increased project control by daily scrum meetings, Transparency and every developer can review the other one work and give feedback frequently. It involves both team and individual interaction over tools and processes. Statistics say that by using agile Methodology, on average, the time to market is 37% faster and the efficiency of your team is increased with a productivity higher by 16% than the average.

Figure 1: Scrum Meeting

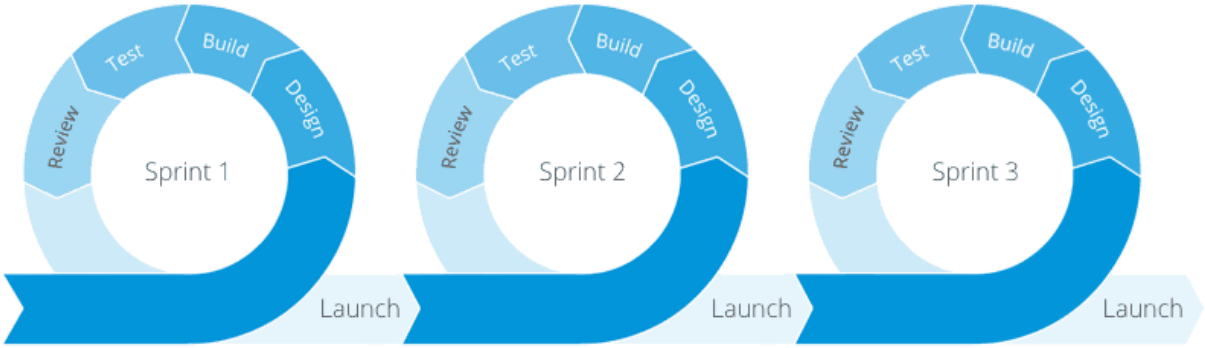

Figure 2: Sprints

Phase One: Requirement gathering

Requirements gathering are based on the assigned term project document following it a brainstorming operation to simulate the user role like how to use the system and the expected input/output and online meetings with the team held to do such things, these requirements are analyzed for their validity and the possibility of incorporating and wrong user input, Good gathering and analysis of requirements is important as it sets clear targets for everyone to aim for.
It can be a lot of hard work, but it need not be a horrifying task if you can keep some key points in mind.

Phase Two: Designing

After requirement gathering and analysis we will have an upper view and strong base to start on the design phase to transform the user requirements as described in the documents into a form implementable using a programming language, this leads to improve the system integration, by applying the design instruction, which helps developers to perform system tasks more effectively.

Phase Three: Development

The developing phase is the next phase every developer is assigned to a specific task/function and using design documents he starts to code, this phase is the core of the whole system and takes the major part of time and not finished till the testing is done and the code is approved .

Phase Four: Testing

Testing phase comes after developing to validate the requirements to meet the wanted output .There are 2 phases involved in Testing Life Cycle: unit testing, integration testing, system testing and Acceptance Testing.
Unit Testing: During this first round of testing focus on specific units of the system to determine whether each one is fully functional.
Integration Testing: Integration testing allows developers the opportunity to combine all of the units within a program and test them as a group.

System Testing: the first level in which the complete application is tested as a whole.

Acceptance Testing: The final level, Acceptance testing (or User Acceptance Testing), is conducted to determine whether the system is ready for release

There is must be a test cases Preparation and test scenarios

Phase Five: Reviewing

The objective of software review is:

- To improve the productivity of the developers.
- To make the testing process time and cost effective.
- To make the final system with fewer defects.
- To eliminate the weakness points

The selected review is peer review.

Peer review is the process of assessing the technical content and quality of the product and it is usually conducted by the author of the work product along with some other developers.

Peer review is performed in order to examine or resolve the defects in the software, whose quality is also checked by other developers.

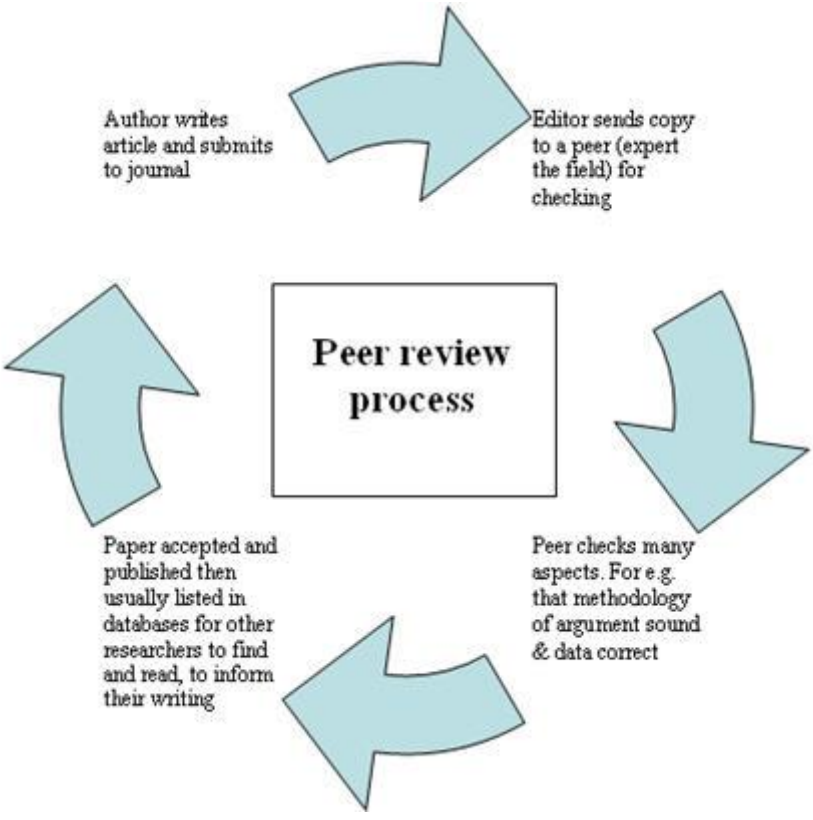

Figure 3: Peer review process

6. System Architecture

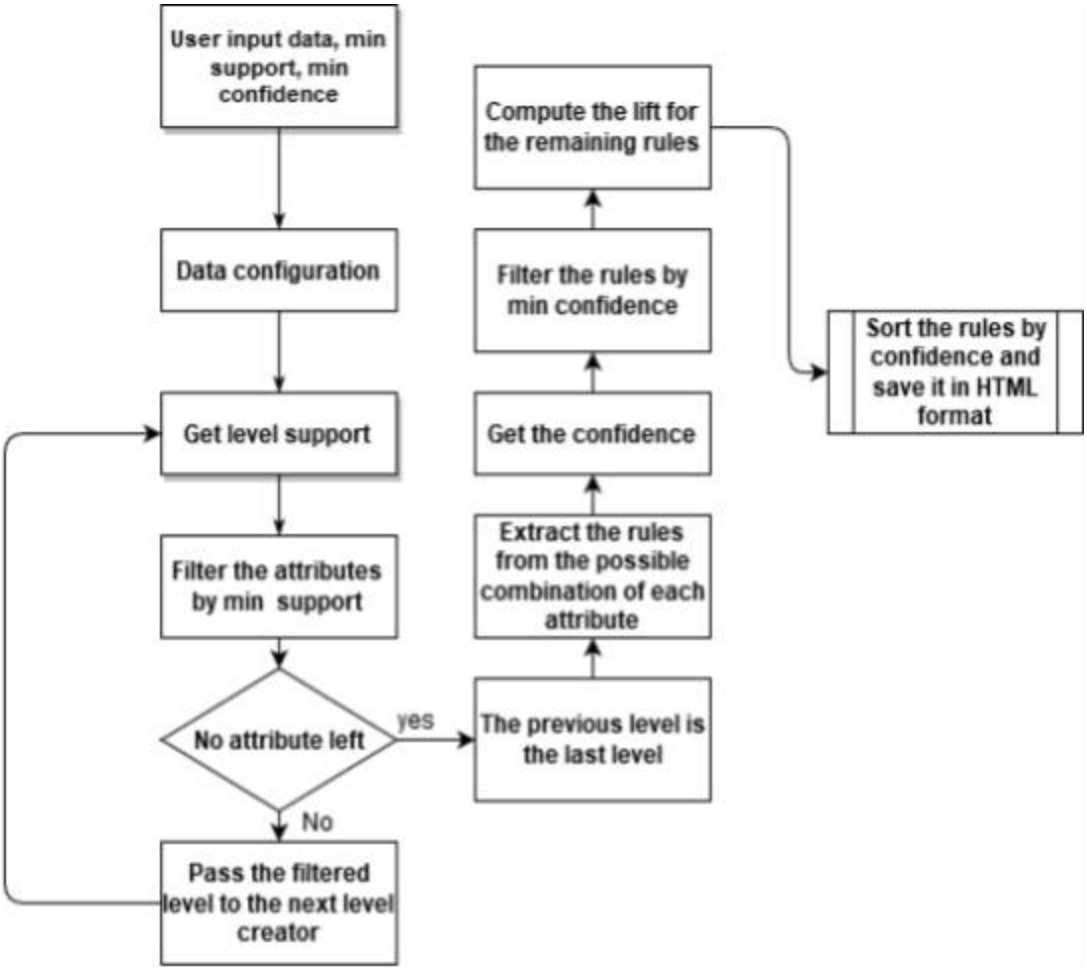

Figure 4: The flow diagram represents the outer structure of the system

7. Development Environment

For data mining R is the optimal choice, but we are not familiar with it, so we chose Python as it's the second commonly used language in data mining.

We used Pandas package to re-data converting the text data to a csv file and then to a dataframe also Pandas let us manipulate the data frame columns and merge them to make a new combination of columns.

Spyder IDLE was chosen, and that is because it has a variable explorer window which helps us to debug through the progress.

Figure 5: Development Tools

8. Testing Cases and Result

- **Test Case 1**

  Minimum support = 322 appearance, Minimum confidence = 70%

  - Sorted according to Lift

	Rule	Confidence	Lift	Leverage
0	9[1_car] --> 0[Social_class_D]^0[2_cars]^0[No_car]	78.415842	11.528713	0.062118
1	0[Social_class_D]^9[1_car] --> 0[2_cars]^0[No_car]	100.000000	11.528713	0.062118
2	0[2_cars]^0[No_car] --> 0[Social_class_D]^9[1_car]	78.415842	11.528713	0.062118
3	0[Social_class_D]^0[2_cars]^0[No_car] --> 9[1_car]	100.000000	11.528713	0.062118
4	0[Unskilled_labourers]^0[Rented_house] --> 0[Social_class_D]^9[Home_owners]	96.482412	7.705358	0.057397
5	0[Unskilled_labourers]^9[Home_owners] --> 0[Social_class_D]^0[Rented_house]	96.482412	7.705358	0.057397
6	0[Rented_house]^0[No_car] --> 0[Social_class_D]^9[Home_owners]	87.800370	7.011986	0.069952
7	9[Home_owners]^0[No_car] --> 0[Social_class_D]^0[Rented_house]	87.800370	7.011986	0.069952
8	0[Unskilled_labourers]^0[Social_class_D]^0[Rented_house] --> 9[Home_owners]	100.000000	6.134879	0.055206
9	0[Unskilled_labourers]^0[Social_class_D]^9[Home_owners] --> 0[Rented_house]	100.000000	6.134879	0.055206
10	0[Social_class_D]^0[Rented_house]^0[No_car] --> 9[Home_owners]	100.000000	6.134879	0.068288
11	0[Social_class_D]^9[Home_owners]^0[No_car] --> 0[Rented_house]	100.000000	6.134879	0.068288
12	2[2_cars]^0[No_car] --> 0[Social_class_D]^7[1_car]	81.002088	6.022914	0.055579
13	9[1_car]^0[No_car] --> 0[Social_class_D]^0[2_cars]	78.415842	5.713855	0.056114
14	7[1_car]^0[No_car] --> 0[Social_class_D]^2[2_cars]	81.002088	5.614216	0.054773
15	0[Social_class_D]^2[2_cars]^0[No_car] --> 7[1_car]	100.000000	4.120311	0.050469
16	0[Social_class_D]^9[1_car]^0[2_cars] --> 0[No_car]	100.000000	4.015172	0.051078
17	9[1_car]^0[2_cars] --> 0[Social_class_D]^0[No_car]	78.415842	3.969687	0.050684
18	0[Social_class_D]^7[1_car]^2[2_cars] --> 0[No_car]	88.990826	3.573135	0.047992
19	0[Social_class_D]^7[1_car]^0[No_car] --> 2[2_cars]	100.000000	3.330664	0.046635
20	0[Social_class_D]^9[1_car]^0[No_car] --> 0[2_cars]	100.000000	3.140237	0.046358
21	0[Unskilled_labourers]^0[Rented_house]^9[Home_owners] --> 0[Social_class_D]	96.482412	2.154663	0.035346
22	0[Rented_house]^9[Home_owners]^0[No_car] --> 0[Social_class_D]	87.800370	1.960774	0.039977
23	7[1_car]^2[2_cars]^0[No_car] --> 0[Social_class_D]	81.002088	1.808953	0.029803
24	9[1_car]^0[2_cars]^0[No_car] --> 0[Social_class_D]	78.415842	1.751197	0.029177

Figure 6: Test Case 1 sorted by Lift

- Sorted according to Leverage

	Rule	Confidence	Lift	Leverage
0	0[Rented_house]^0[No_car] --> 0[Social_class_D]^9[Home_owners]	87.800370	7.011986	0.069952
1	9[Home_owners]^0[No_car] --> 0[Social_class_D]^0[Rented_house]	87.800370	7.011986	0.069952
2	0[Social_class_D]^0[Rented_house]^0[No_car] --> 9[Home_owners]	100.000000	6.134879	0.068288
3	0[Social_class_D]^9[Home_owners]^0[No_car] --> 0[Rented_house]	100.000000	6.134879	0.068288
4	9[1_car] --> 0[Social_class_D]^0[2_cars]^0[No_car]	78.415842	11.528713	0.062118
5	0[Social_class_D]^9[1_car] --> 0[2_cars]^0[No_car]	100.000000	11.528713	0.062118
6	0[2_cars]^0[No_car] --> 0[Social_class_D]^9[1_car]	78.415842	11.528713	0.062118
7	0[Social_class_D]^0[2_cars]^0[No_car] --> 9[1_car]	100.000000	11.528713	0.062118
8	0[Unskilled_labourers]^0[Rented_house] --> 0[Social_class_D]^9[Home_owners]	96.482412	7.705358	0.057397
9	0[Unskilled_labourers]^9[Home_owners] --> 0[Social_class_D]^0[Rented_house]	96.482412	7.705358	0.057397
10	9[1_car]^0[No_car] --> 0[Social_class_D]^0[2_cars]	78.415842	5.713855	0.056114
11	2[2_cars]^0[No_car] --> 0[Social_class_D]^7[1_car]	81.002088	6.022914	0.055579
12	0[Unskilled_labourers]^0[Social_class_D]^0[Rented_house] --> 9[Home_owners]	100.000000	6.134879	0.055206
13	0[Unskilled_labourers]^0[Social_class_D]^9[Home_owners] --> 0[Rented_house]	100.000000	6.134879	0.055206
14	7[1_car]^0[No_car] --> 0[Social_class_D]^2[2_cars]	81.002088	5.614216	0.054773
15	0[Social_class_D]^9[1_car]^0[2_cars] --> 0[No_car]	100.000000	4.015172	0.051078
16	9[1_car]^0[2_cars] --> 0[Social_class_D]^0[No_car]	78.415842	3.969887	0.050884
17	0[Social_class_D]^2[2_cars]^0[No_car] --> 7[1_car]	100.000000	4.120311	0.050469
18	0[Social_class_D]^7[1_car]^2[2_cars] --> 0[No_car]	88.990826	3.573135	0.047992
19	0[Social_class_D]^7[1_car]^0[No_car] --> 2[2_cars]	100.000000	3.330664	0.046635
20	0[Social_class_D]^9[1_car]^0[No_car] --> 0[2_cars]	100.000000	3.140237	0.046358
21	0[Rented_house]^9[Home_owners]^0[No_car] --> 0[Social_class_D]	87.800370	1.960774	0.039977
22	0[Unskilled_labourers]^0[Rented_house]^9[Home_owners] --> 0[Social_class_D]	96.482412	2.154663	0.035346
23	7[1_car]^2[2_cars]^0[No_car] --> 0[Social_class_D]	81.002088	1.808953	0.029803
24	9[1_car]^0[2_cars]^0[No_car] --> 0[Social_class_D]	78.415842	1.751197	0.029177

Figure 7: Test Case 1 sorted by Leverage

- **Test Case 2**

  Minimum support = 721 appearance, Minimum confidence = 60%

  - Sorted according to Lift

	Rule	Confidence	Lift	Leverage
0	0[Rented_house] --> 0[Social_class_D]^9[Home_owners]	76.817703	6.134879	0.104804
1	9[Home_owners] --> 0[Social_class_D]^0[Rented_house]	76.817703	6.134879	0.104804
2	0[Social_class_D]^0[Rented_house] --> 9[Home_owners]	100.000000	6.134879	0.104804
3	0[Social_class_D]^9[Home_owners] --> 0[Rented_house]	100.000000	6.134879	0.104804
4	0[Rented_house]^9[Home_owners] --> 0[Social_class_D]	76.817703	1.715507	0.052225

Figure 8: Test Case 2 sorting by Lift

  - Sorted according to Leverage

	Rule	Confidence	Lift	Leverage
0	0[Rented_house] --> 0[Social_class_D]^9[Home_owners]	76.817703	6.134879	0.104804
1	9[Home_owners] --> 0[Social_class_D]^0[Rented_house]	76.817703	6.134879	0.104804
2	0[Social_class_D]^0[Rented_house] --> 9[Home_owners]	100.000000	6.134879	0.104804
3	0[Social_class_D]^9[Home_owners] --> 0[Rented_house]	100.000000	6.134879	0.104804
4	0[Rented_house]^9[Home_owners] --> 0[Social_class_D]	76.817703	1.715507	0.052225

Figure 9: Test Case 2 sorting by Leverage

- **Test Case 3**

  Minimum support = 1030 appearance, Minimum confidence = 30%

  - Sorted according to Lift

	Rule	Confidence	Lift	Leverage
0	0[No_car] --> 0[Social_class_D]	79.310345	1.771173	0.086004
1	0[Social_class_D] --> 0[No_car]	44.112006	1.771173	0.086004

Figure 10: Test Case 3 sorting by Lift

  - Sorted according to Leverage

	Rule	Confidence	Lift	Leverage
0	0[Social_class_D] --> 0[No_car]	44.112006	1.771173	0.086004
1	0[No_car] --> 0[Social_class_D]	79.310345	1.771173	0.086004

Figure 11: Test Case 3 sorting by Leverage

9. Conclusion

To summarize, this is the most simple and easy-to-understand algorithm among association rule learning algorithms, the resulting rules are intuitive and easy to understand by the end user. It doesn't require labeled data as it is fully unsupervised, as a result, you can use it in many different situations because unlabeled data is often more accessible.

Apriori algorithm is a classic algorithm, is useful in mining frequent itemsets and relevant association rules. Usually, you operate this algorithm on a database containing a large number of transactions. One such example is the items customers buy at a market.
It is very important for effective Market Basket Analysis and it helps the customers in purchasing their items with more ease which increases the sales of the markets. It has also been used in the field of healthcare for the detection of adverse drug reactions.

10. The progress in the project using Earned Value method

- Initial Timeline

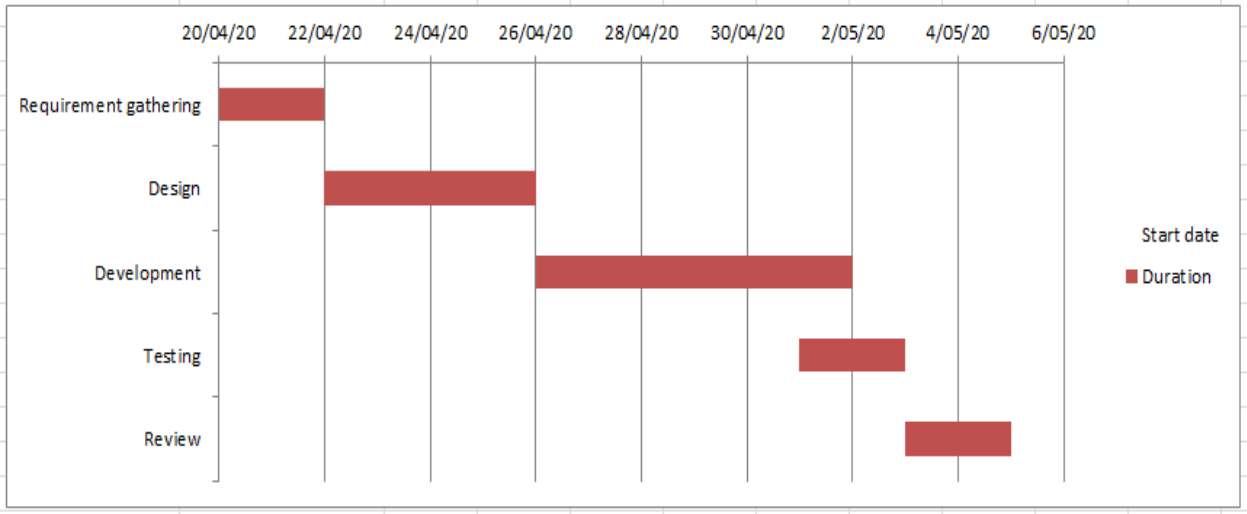

Figure 12: Initial Timeline using Earned Value Method

- Actual Timeline

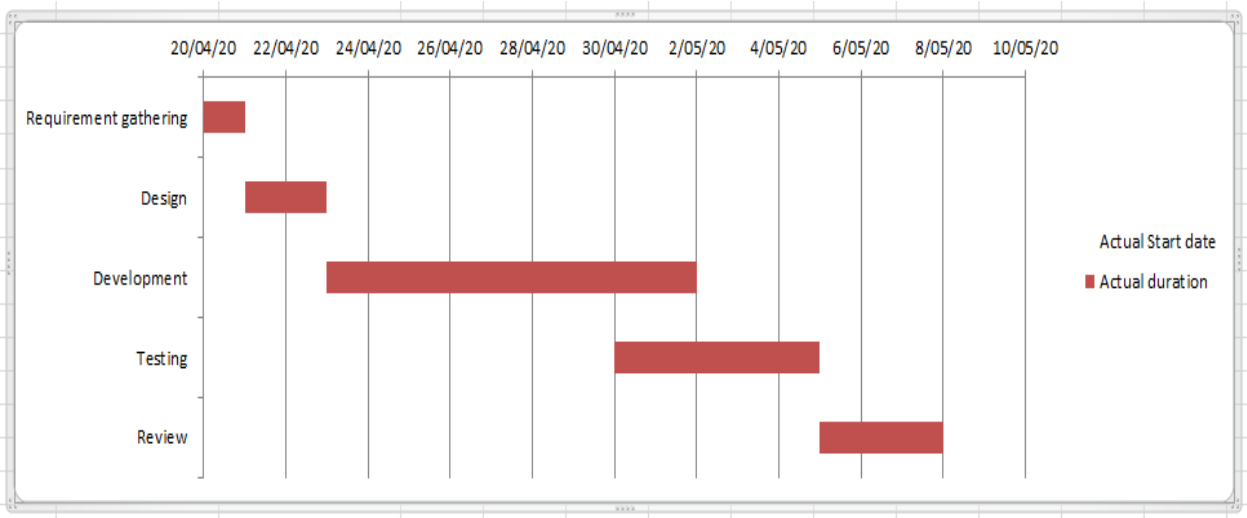

Figure 13: Actual Timeline using Earned Value Method

Thank

You

Team 26