Ain Shams University
Faculty of Engineering, Computer and Engineering Department
CSE412: Selected Topics in Computer Engineering

Apriori Algorithm Project

Team Members

Name	ID	Sec
Ahmed Ali Ahmed Mostafa	1300156	1
Eslam Ahmed Mohamed Ahmed	1500259	1
Eslam Tarek Fawzy Omran	1500269	1
Eslam Shabaan Hussien Ali	15T0055	1
Baher Abdelfatah Atia Abdelfatah	1500405	1
Amr Samy Ahmed Ahmed	1500955	2

Group number: 8

(column from 10 to 21)

Abstract

Figure 1: Usage of Apriori Algorithm in Business

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.

Table of Contents

1. Introduction

1.1 Purpose

This report is intended to describe the project in details, the steps, timeline of the project, what we have learned, the algorithm and the architecture (design) …etc.

This report is intended for those who have a background in data science and good knowledge in Computer Engineering field.

1.2 List of definitions

1.2.1 Transactional Database:

A database that consists of a set of records where each record represents a transaction. A transaction typically carries a unique identification number (transaction ID or TID) and a list of the items making up the transaction (for example, items purchased in a store).

1.2.2 Apriori algorithm:

It is an algorithm used to find the relation between items or categories.

1.2.3 Support count:

Frequency of occurrence of an itemset.

1.2.4 Support:

Fraction of transactions that contain an itemset.

1.2.5 Frequent itemset:

An itemset whose support is greater than or equal to a minimum support threshold.

1.2.6 Apriori principle:

If an itemset is frequent all of its subsets must also be frequent. Confidence: The % of transactions that contain X, which also contain Y.

1.2.7 Lift:

measures how many times more often X and Y occur together than expected if they were statistically independent. It is a measure of how X and Y are really related rather than coincidentally happening together.

1.2.8 Leverage:

measures the difference in the probability of X and Y appearing together in the data set compared to what would be expected if X and Y were statistically independent.

1.3 Overview

The rest of the report will focus on: beneficiaries, project aims, detailed project description and we will go through project phases, system architecture, development environment, test cases and results conclusion, use earned value method to assess the progress in our project, we intend to have a brief description for each.

2. Beneficiaries

The reader will learn how to apply Apriori algorithm which is widely used in data mining and recommendation marketing.

Taking a real-world example, our dataset itself represents a real customer data for an insurance company, we get the association between customer attributes.

In general Apriori algorithm is useful for data Mining companies and data Analysts, businesses companies who are interested in finding an association between the items they are selling which can later be used to alter product placement, organization, and advertisement.

3. Project aims and objectives

This Project makes us learn how to implement Apriori algorithm from scratch a very vital algorithm needed in data mining and data science generally, Learn widely used Programming language in Data Analysis (R) to deal with real data and see real results.

4. Detailed Project Description

Apriori algorithm is used for finding frequent item sets in a dataset, We apply an iterative approach or level-wise search where k-frequent item sets are used to find k+1 item sets. Apriori uses a bottom-up approach, where frequent subsets are extended one item at a time (a step is known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

C_k: Candidate item set of size k
L_k : Frequent item set of size k
L_1= {frequent items};
For (k=1; L_k!=Φ;k++) do begin
C_{k+1} = candidates generated from L_k;
For each transaction t in database do
Increment the count of all candidates in C_{k+1}
Those are contained in t
L_{k+1} = candidates in C_{k+1} with min_support
End
Return U_k L_k;

Figure 2: Apriori Algorithm Pseudo Code

The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database. In our project, we use this algorithm to mine data out of a dataset that represents customer data for an insurance company, it has 86 attributes with 5822 records, we are only interested in 12 attributes of them.

We will be explaining the used functions in our Implementation so the reader can have a deeper understanding on how we tackled Apriori.

R Code of Apriori:-

[1] getMinSupport():

Input: none

Output: minSupport

Description: Ask user to enter the min support

[2] getMinConfidence()

Input: none

Output: minConfidence

Description: Ask user to enter the min confidence

[3] readData()

Input: none

Output: dataframe -> containing the columns from 10 to 21

Description: Read dataset from ticdata2000.txt taking only columns from 10 to 21

[4] calculateTotalSupport(data,minSupport)

Input:

- dataframe -> containing the columns from 10 to 21
- minSupport

Output: supportOfAllValidItemsets -> list of valid item sets

Description: Applying apriori algorithim to find all frequent item sets

[5] GetRules(supportOfAllValidItemsets)

Input: supportOfAllValidItemsets -> list of valid item sets

Output: rules -> all possible rules of most strong item sets

Description: Find all possible rules exceeding minSupport

[6] filteringByConfidence(rules, itemsetsSupport, minConfidence)

Input:

- rules -> all possible rules of most strong item sets
- itemsetsSupport -> list of valid item sets
- minConfidence

Output: rules -> all possible rules with minConfidence or higher

Description: Find all possible rules exceeding minConfidence

[7] calcLiftLeverage(filteredRules, allFreq)

Input:

- filteredRules -> all possible rules with minConfidence or higher
- allFreq -> contains support of all frequent item sets

Output:dataframe -> contain lift and leverage of each rule

Description: calculate lift and leverage of association rules

5. Project Phases

- Phase 1: Download our dataset.
- Phase 2: Understand the dataset and its attributes.
- Phase 3: Analyze the project requirements.
- Phase 4: Study and discuss the Apriori Algorithm.
- Phase 5: Design the schedule for our project.
- Phase 6: Divide the code into separate functions.
- Phase 7: Every member started to implement his function.
- Phase 8: Code Integration.
- Phase 9: Code Testing.
- Phase 10: Report.
- Phase 11: Demo and Presentation.

6. System Architecture

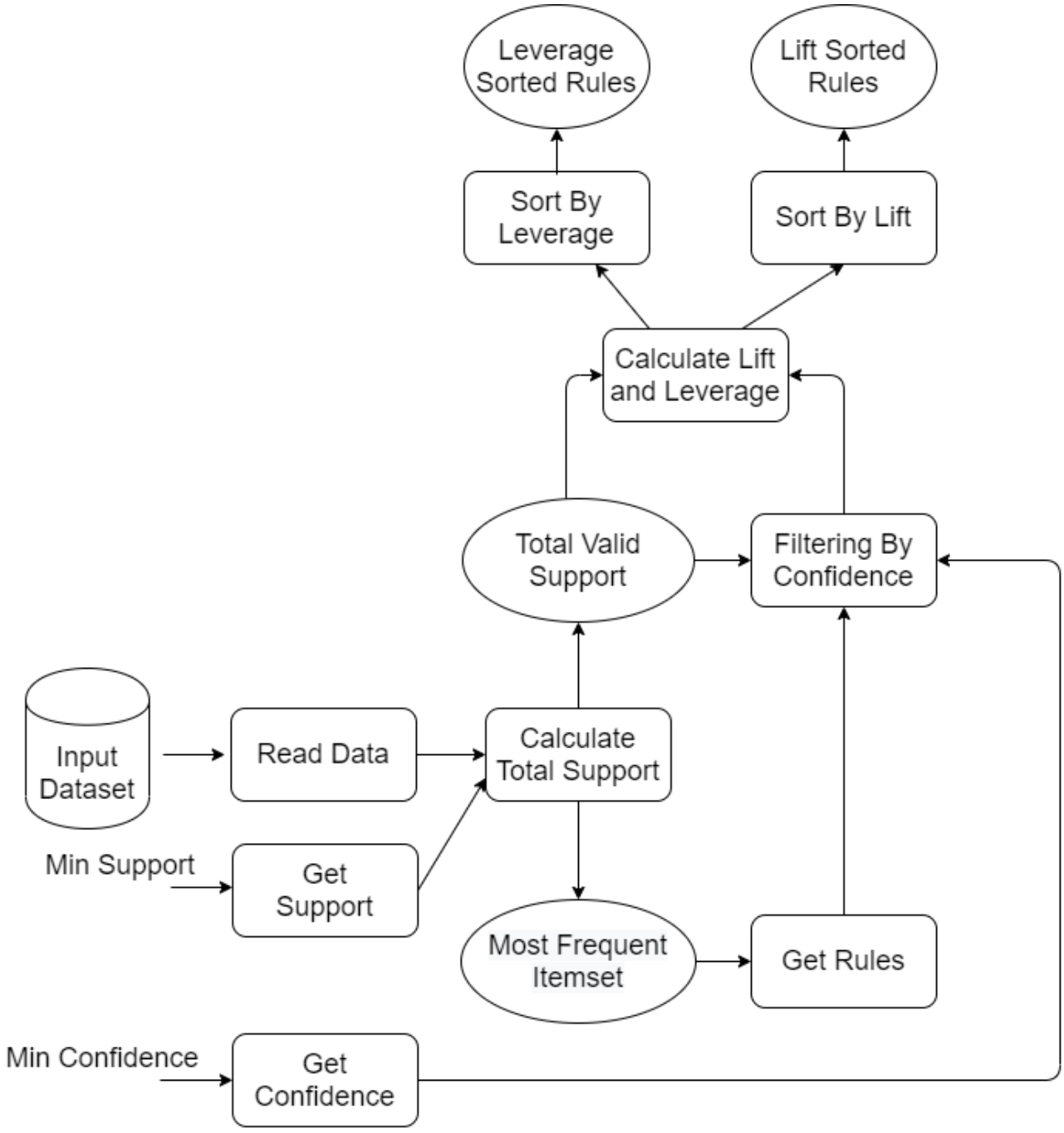

Figure 3: System Architecture

This diagram represents the pipeline of our implementation, For detailed description of the functions, Please refer to Detailed Project Description.

7. Development Environment

- Programming Language: R 3.6.3
- Integrated Development Environment: Rstudio 1.0.136
- Operating System: Windows 10 operating system
- Libraries: library(sqldf), library(gtools), library(stringr)

8. Testing Cases and Results

TestCase 1:

MinSupport: 0.5

MinConfidence: 0.5

Sorted by Leverage:

Entrepreneur_0 -> Farmer_0	0.0337045861060069
Farmer_0 -> Entrepreneur_0	0.0337045861060069

Sorted by Lift:

Entrepreneur_0 -> Farmer_0	1.06558925843559
Farmer_0 -> Entrepreneur_0	1.06558925843559

TestCase 2:

MinSupport: 0.3

MinConfidence: 0.6

Sorted by Leverage:

High level education_0 -> Entrepreneur_0	0.0602619790767462
Living together_0 -> Entrepreneur_0	0.0354178425784239
Entrepreneur_0 -> Farmer_0	0.0337045861060069
Farmer_0 -> Entrepreneur_0	0.0337045861060069
Living together_0 -> Farmer_0	0.00620067144831776

Sorted by Lift:

High level education_0 -> Entrepreneur_0	1.22809489123394
Living together_0 -> Entrepreneur_0	1.11757497692596
Entrepreneur_0 -> Farmer_0	1.06558925843559
Farmer_0 -> Entrepreneur_0	1.06558925843559
Living together_0 -> Farmer_0	1.02055943705707

TestCase 3:

MinSupport: 0.17

MinConfidence: 0.6

Sorted by Leverage:

High status_0 -> High level education_0, Entrepreneur_0	0.0988534115434873
High status_0, Entrepreneur_0 -> High level education_0	0.098464777993564
High level education_0, Farmer_0 -> Entrepreneur_0	0.0483363014595014
High level education_0, High status_0 -> Entrepreneur_0	0.0396893893629643
High status_0 -> Entrepreneur_0, Farmer_0	0.0368414456542609
High level education_0 -> Entrepreneur_0, Farmer_0	0.0366454324981316
High status_0, Farmer_0 -> Entrepreneur_0	0.034605733284509
Singles_0 -> Entrepreneur_0, Farmer_0	0.0271221551392797
Singles_0, Entrepreneur_0 -> Farmer_0	0.026667702000054
Living together_0, Farmer_0 -> Entrepreneur_0	0.025966078749141
High status_0, Entrepreneur_0 -> Farmer_0	0.0142261769964577
Singles_0, Farmer_0 -> Entrepreneur_0	0.0106229748896644
High level education_0, Entrepreneur_0 -> Farmer_0	0.00585012534339177
Living together_0, Entrepreneur_0 -> Farmer_0	0.00500388191015702

Sorted by Lift:

High status_0 -> High level education_0, Entrepreneur_0	2.16390930223187
High status_0, Entrepreneur_0 -> High level education_0	2.15405277092844
High level education_0, High status_0 -> Entrepreneur_0	1.27543665433381
High status_0 -> Entrepreneur_0, Farmer_0	1.25702643113776
High level education_0, Farmer_0 -> Entrepreneur_0	1.2540786301522
High status_0, Farmer_0 -> Entrepreneur_0	1.23772098657295
High level education_0 -> Entrepreneur_0, Farmer_0	1.18147378472719
Singles_0", "->", "Entrepreneur_0", "Farmer_0"	1.16412643028888
Singles_0, Entrepreneur_0 -> Farmer_0	1.16093378341808
Living together_0, Farmer_0 -> Entrepreneur_0	1.11775316171182
High status_0, Entrepreneur_0 -> Farmer_0	1.08572441383861
Singles_0, Farmer_0 -> Entrepreneur_0	1.05844802921549
High level education_0, Entrepreneur_0 -> Farmer_0	1.02513721302398
Living together_0, Entrepreneur_0 -> Farmer_0	1.02072215380405

TestCase 4:

MinSupport: 0.1

MinConfidence: 0.8

Sorted by Leverage:

Married_9, Singles_0, Farmer_0 -> Living together_0, Other relation_0	0.0872224322129036
Living together_0, Other relation_0, Singles_0, "Farmer_0 -> Married_9	0.0872224322129036
Married_9, Singles_0 -> Living together_0, Other relation_0, Farmer_0	0.0870618807987471
Married_9, Farmer_0 -> Living together_0, Other relation_0, Singles_0	0.0870618807987471
Living together_0, Other relation_0, Singles_0 -> Married_9, Farmer_0	0.0870618807987471
Living together_0, Other relation_0, Farmer_0 -> Married_9, Singles_0	0.0870618807987471
Married_9, Other relation_0, Farmer_0 -> Living together_0, Singles_0	0.0810424123614086
Married_9, Living together_0, Singles_0, Farmer_0 -> Other relation_0	0.0806477898484066
Married_9, Living together_0, Singles_0 -> Other relation_0, Farmer_0	0.080351291922594
Married_9, Living together_0, Farmer_0 -> Other relation_0, Singles_0	0.0798343529518389
Married_9, Other relation_0, Singles_0 -> Living together_0, Farmer_0	0.0656274704472699
Married_9, Living together_0, Other relation_0, Farmer_0 -> Singles_0	0.064400352564061
Married_9, Other relation_0, Singles_0, Farmer_0 -> Living together_0	0.0585299296512205
Married_9, Living together_0, Other relation_0, Singles_0 -> Farmer_0	0.018574400209773

Sorted by Lift:

Married_9, Singles_0, Farmer_0 -> Living together_0, Other relation_0	7.33249370277078
Living together_0, Other relation_0, Singles_0, "Farmer_0 -> Married_9	7.33249370277078
Married_9, Singles_0 -> Living together_0, Other relation_0, Farmer_0	7.24800873994842
Married_9, Farmer_0 -> Living together_0, Other relation_0, Singles_0	7.24800873994842
Living together_0, Other relation_0, Singles_0 -> Married_9, Farmer_0	7.24800873994842
Living together_0, Other relation_0, Farmer_0 -> Married_9, Singles_0	7.24800873994842
Married_9, Other relation_0, Farmer_0 -> Living together_0, Singles_0	5.06150088416023
Married_9, Living together_0, Singles_0, Farmer_0 -> Other relation_0	4.96334185848252
Married_9, Living together_0, Singles_0 -> Other relation_0, Farmer_0	4.8920594363323
Married_9, Living together_0, Farmer_0 -> Other relation_0, Singles_0	4.77255693604872
Married_9, Other relation_0, Singles_0 -> Living together_0, Farmer_0	2.85552130044843
Married_9, Living together_0, Other relation_0, Farmer_0 -> Singles_0	2.7597711138449
Married_9, Other relation_0, Singles_0, Farmer_0 -> Living together_0	2.37826797385621
Married_9, Living together_0, Other relation_0, Singles_0 -> Farmer_0	1.22535779942615

9. Conclusion

Apriori algorithm is efficient as it tests all attributes as candidates and all their combinations to get the highest level of combinations which is computationally expensive. It calculates some parameters as lift and leverage to prioritize the rules, also gets support and confidence as input to compare them with combinations.

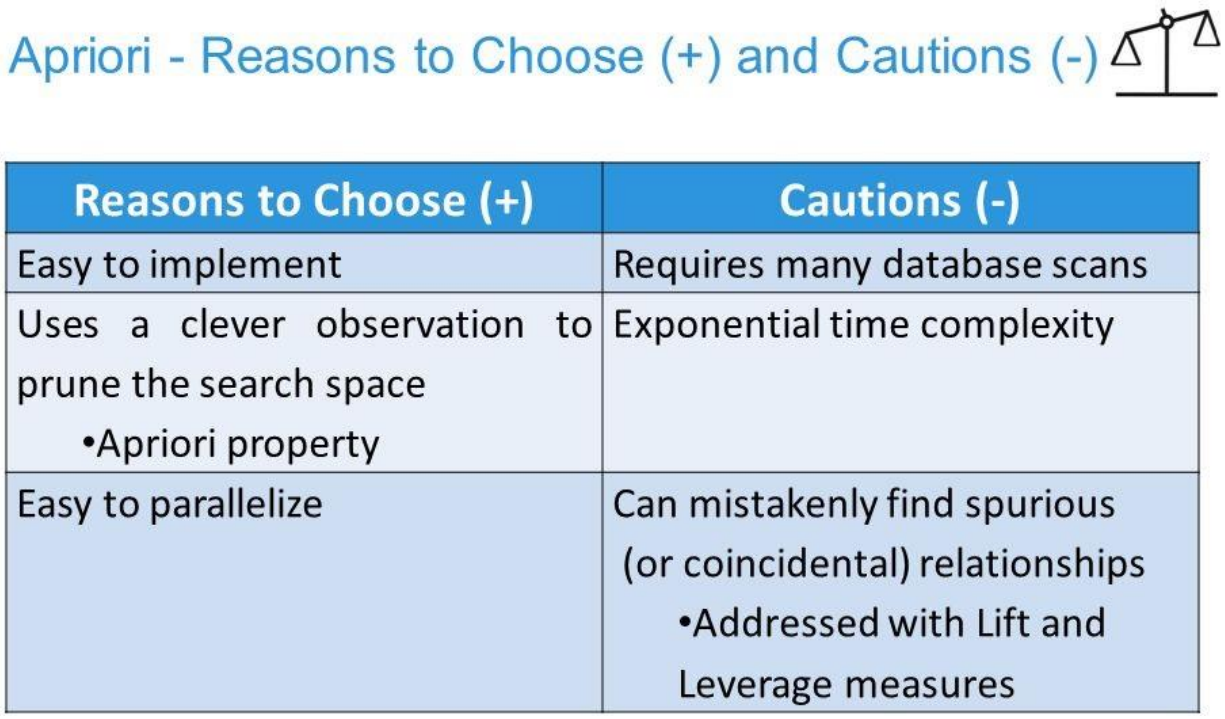

Figure 4: Apriori Algorithm Pros and Cons.

10. Earned Value method

Table 1: Earned value table

Task	Estimated Start	Estimated End	Estimated duration	Actual duration	Resources
Project discussion	10 April	14 April	5 days	6 days	All team
Project plan	16 April	19 April	3 days	4 days	All team
Function Implementation	20 April	26 April	6 days	7 days	All team
Code Integration	27 April	30 April	4 days	4 days	All team
Code Testing	1 May	3 May	3 days	3 days	All team
Report	4 May	6 May	3 days	3 days	Eslam Tarek, Eslam Shabaan, Baher Abdelfatah
Demo and Presentation	7 May	9 May	3 days	3 days	Ahmed Ali, Eslam Ahmed, Amr Samy

Note:

Actual timeline of the project is delayed 3 days from estimated timeline.

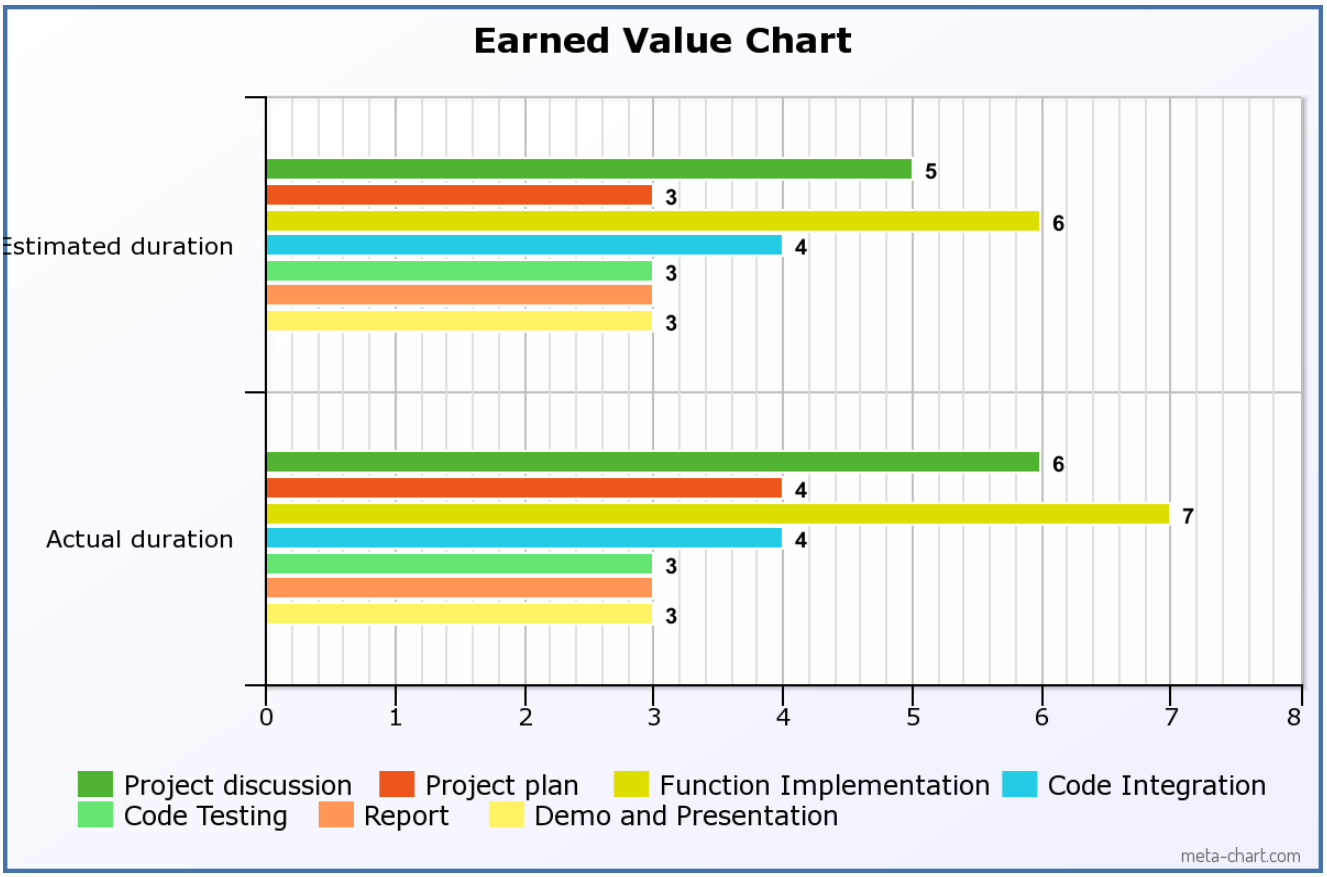

Figure 5: Bar chart for time analysis