Market Basket Analysis

Algorithms for Massive Data

MSc in Data Science and Economics

Nazli Begum Cirpanli
942345

September, 2021

Contents

1 Aim of the Analysis

This analysis aims at uncovering frequent actors in the IMDB dataset taken fromhttps://www.kaggle.com/.As a well-known method for finding frequent itemsets, market basket analysis has been employed by considering movies as baskets and actors as items. Actors that appear in movies frequently as either singletons or together is what this analysis tries to find.

Apriori and FP Growth algorithms have been performed and compared on the same dataset. Results of the experiments have shown that the same itemsets were found by both algorithms as frequent; however there is a significant difference with respect to the execution time of these algorithms.

2 Dataset

IMDB Dataset contains 5 different tables, of which each contains different information about movie titles. In order to create baskets, movie ids/names and actor ids/names are primarily necessary. After having seen the tables; names, basics and principals tables are selected as useful for the purpose of this analysis, therefore have been used as inputs for finding frequent itemsets.

2.1 Tables

Selected tables and their content can be seen below.

- NAMES
  Contains the following information for people:

nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0050419,tt0053137,tt0072308,tt0043044
nm0000002	Lauren Bacall	1924	2014	actress,soundtrack	tt0071877,tt0117057,tt0038355,tt0037382
nm0000003	Brigitte Bardot	1934	\N	actress,soundtrack,producer	tt0054452,tt0049189,tt0059956,tt0057345
nm0000004	John Belushi	1949	1982	actor,writer,soundtrack	tt0077975,tt0072562,tt0080455,tt0078723
nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0069467,tt0050976,tt0083922,tt0050986

only showing top 5 rows

Figure 1: Names table

- nconst (string) – unique identifier of the person
- primaryName (string)– name of the person
- birthYear – (YYYY) birth year of the person
- deathYear – (YYYY) death year of the person

- primaryProfession (array of strings)– the top-3 professions of the person
- knownForTitles (array of tconsts) – title IDs of the person is known for

- BASICS
  Contains basic information about movie titles and has 9 columns:

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
tt0000001	short	Carmencita	Carmencita	0	1894	\N	1	Documentary,Short
tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	\N	5	Animation,Short
tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	\N	4	Animation,Comedy,Romance
tt0000004	short	Un bon bock	Un bon bock	0	1892	\N	\N	Animation,Short
tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	\N	1	Comedy,Short

only showing top 5 rows

Figure 2: Basics table

- tconst (string) – unique identifier of the title
- titleType (string) – the type of the title
- primaryTitle (string) – the more popular title
- originalTitle (string) - original title, in the original language
- isAdult (boolean) – 0 for non-adult title ; 1 for adult title
- startYear (YYYY) –the release year of a title
- endYear (YYYY) –the end year of a title
- runtimeMinutes –runtime of the title, in minutes
- genres (string array) –genres of the title

- PRINCIPALS
  Contains information about cast/crew for titles:

tconst	ordering	nconst	category	job	characters
tt0000001	1	nm1588970	self	\N	["Herself"]
tt0000001	2	nm0005690	director	\N	\N
tt0000001	3	nm0374658	cinematographer	director of photography	\N
tt0000002	1	nm0721526	director	\N	\N
tt0000002	2	nm1335271	composer	\N	\N

only showing top 5 rows

Figure 3: Principals table

- ordering (integer) – row number of a unique title
- nconst (string) – unique identifier number of the person.
- category (string) - the category of the person's job in the title that person appears
- job (string) - the job title
- characters (string) - the name of the character if the person is an actor/actress

2.2 How Data Have Been Organized

Tables have various different attributes from which necessary ones should be selected and used to create movie baskets and actor sets. In order to apply aforementioned algorithms data must be in a specific format. Therefore, data have been organized in a way that can be used in the market basket analysis. Basket data should be in sequential order and the items in each basket should be set of lists.

Firstly, from the names table, people whose primary profession is either actor or actress are filtered. In the primaryProfession column there are many others besides actor or actress.

After filtering operation, new table consists of only people whose profession is either actor or actress. The number of all actors and actresses is 2,914,287.

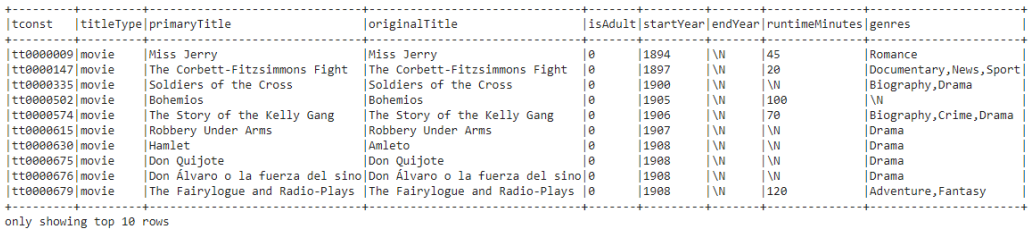
nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
nm0000084	Li Gong	1965	\N	actress	tt0473444,tt0101640,tt0397535,tt0430357
nm0000109	Yasmine Bleeth	1968	\N	actress	tt0131857,tt0115285,tt0337851,tt0119271
nm0000124	Jennifer Connelly	1970	\N	actress	tt0315983,tt0180093,tt0268978,tt0102803
nm0000143	Erika Eleniak	1969	\N	actress	tt0083866,tt0094761,tt0106400,tt0105690
nm0000157	Linda Hamilton	1956	\N	actress	tt0103064,tt6450804,tt0088247,tt0118928
nm0000266	Ursula Andress	1936	\N	actress	tt0061452,tt0055928,tt0082186,tt0060177
nm0000282	Scott Bairstow	1970	\N	actor	tt0283084,tt0182587,tt0120512,tt0119925
nm0000283	Brenda Bakke	1963	\N	actress	tt0114608,tt0107144,tt0119488,tt0114781
nm0000314	Charles Bronson	1921	2003	actor	tt0064116,tt0054047,tt0057115,tt0071402
nm0000319	Yancy Butler	1970	\N	actress	tt0107076,tt0274298,tt1250777,tt1650554

only showing top 10 rows

Figure 4: People whose primary profession are actor or actress

Secondly, since this analysis is only concerned with movies, from the Basics table, rows which have the titleType as 'movie' are filtered. As seen below, table contains 10 different types and titles with title type movie have been selected. The number of movies is 3,800,079.

Lastly, Principals table has been used for bringing together movies and actors since it has both movie and actor ids. Columns which are thought



tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
tt0000009	movie	Miss Jerry	Miss Jerry	0	1894	\N	45	Romance
tt0000147	movie	The Corbett-Fitzsimmons Fight	The Corbett-Fitzsimmons Fight	0	1897	\N	20	Documentary,News,Sport
tt0000335	movie	Soldiers of the Cross	Soldiers of the Cross	0	1900	\N	\N	Biography,Drama
tt0000502	movie	Bohemios	Bohemios	0	1905	\N	100	\N
tt0000574	movie	The Story of the Kelly Gang	The Story of the Kelly Gang	0	1906	\N	70	Biography,Crime,Drama
tt0000615	movie	Robbery Under Arms	Robbery Under Arms	0	1907	\N	\N	Drama
tt0000630	movie	Hamlet	Amleto	0	1908	\N	\N	Drama
tt0000675	movie	Don Quijote	Don Quijote	0	1908	\N	\N	Drama
tt0000676	movie	Don Álvaro o la fuerza del sino	Don Álvaro o la fuerza del sino	0	1908	\N	\N	Drama
tt0000679	movie	The Fairylogue and Radio-Plays	The Fairylogue and Radio-Plays	0	1908	\N	120	Adventure,Fantasy

only showing top 10 rows

Figure 5: Titles whose type are movie

to be necessary have been collected in one table: movie id, movie title, actor/actress id, actor/actress name. 'primaryName' attribute has null values,

tconst	primaryTitle	nconst	primaryName
tt0083109	The Cabbage Soup	nm0000086	Louis de Funès
tt0062120	The Little Bather	nm0000086	Louis de Funès
tt0057422	Squeak-squeak	nm0000086	Louis de Funès
tt0058089	Fantomas	nm0000086	Louis de Funès
tt1253586	Louis de Funès intime	nm0000086	Louis de Funès
tt0067274	Jo	nm0000086	Louis de Funès
tt0079200	The Troops & Aliens	nm0000086	Louis de Funès
tt0048994	Babies Galore	nm0000086	Louis de Funès
tt8768374	Le gendarme et l'empereur	nm0000086	Louis de Funès
tt0066423	Perched on a Tree	nm0000086	Louis de Funès

only showing top 10 rows

Figure 6: Movies and actors

however, since 'tconst' (actor id) will be enough for this analysis, primaryName null values do not pose any threat and will be left untouched.

After checking the table for null values, movie baskets table has been created with movie ids and actor ids.Each movie basket contains actors/actresses who appear in that particular movie. The total number of baskets is 522,758.

3 Exploratory Analysis

Before proceeding with the algorithms and their implementations, some exploratory analysis has been performed with the aim of getting more familiar with the data.

People are categorized by their profession to see how much of the data contains actors and actresses. As seen from the pie chart below, big part of the data consists of actresses and actors. These actors and actresses can appear both in movies and any other kind of titles. That is why the total number of them is greater than the number given before.

Moreover, when the baskets are grouped together according to their sizes, the most common basket size is 10 and there is not any basket with a larger itemset.

Actors have been ranked by the number of their appearances in movies, and top ten most appeared actors can be seen in Figure 8.

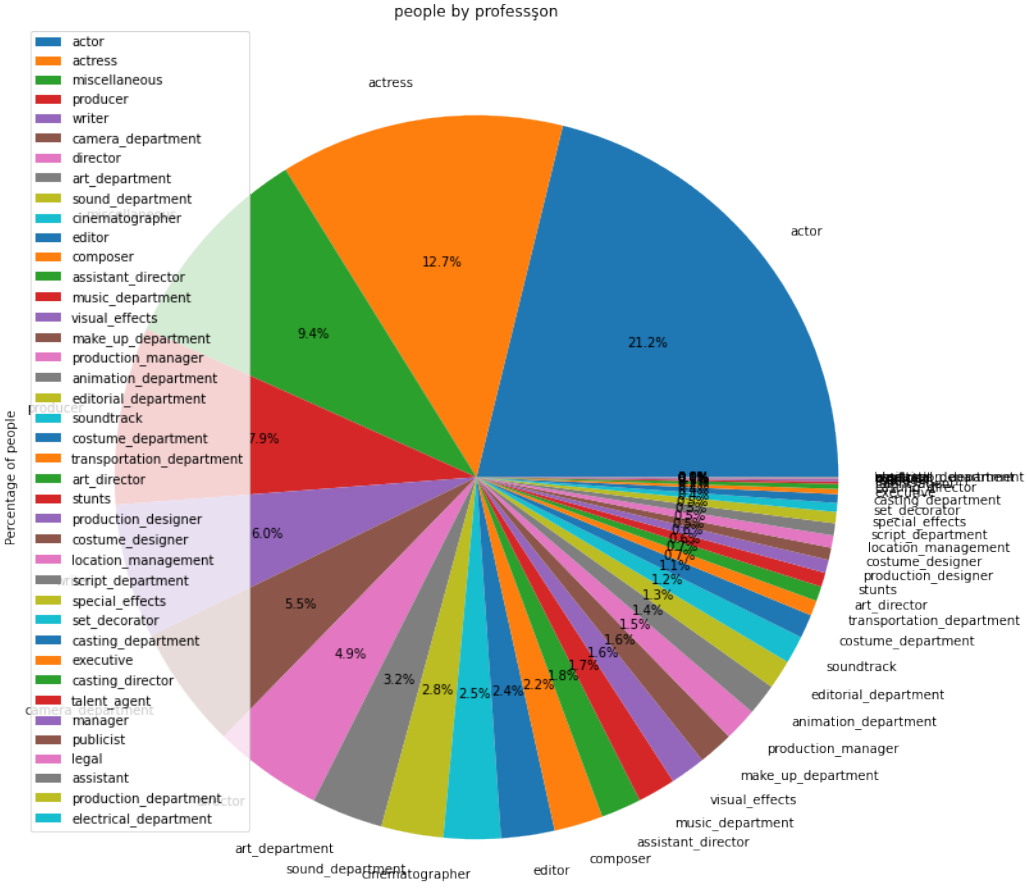


Figure 7: People by their primary profession

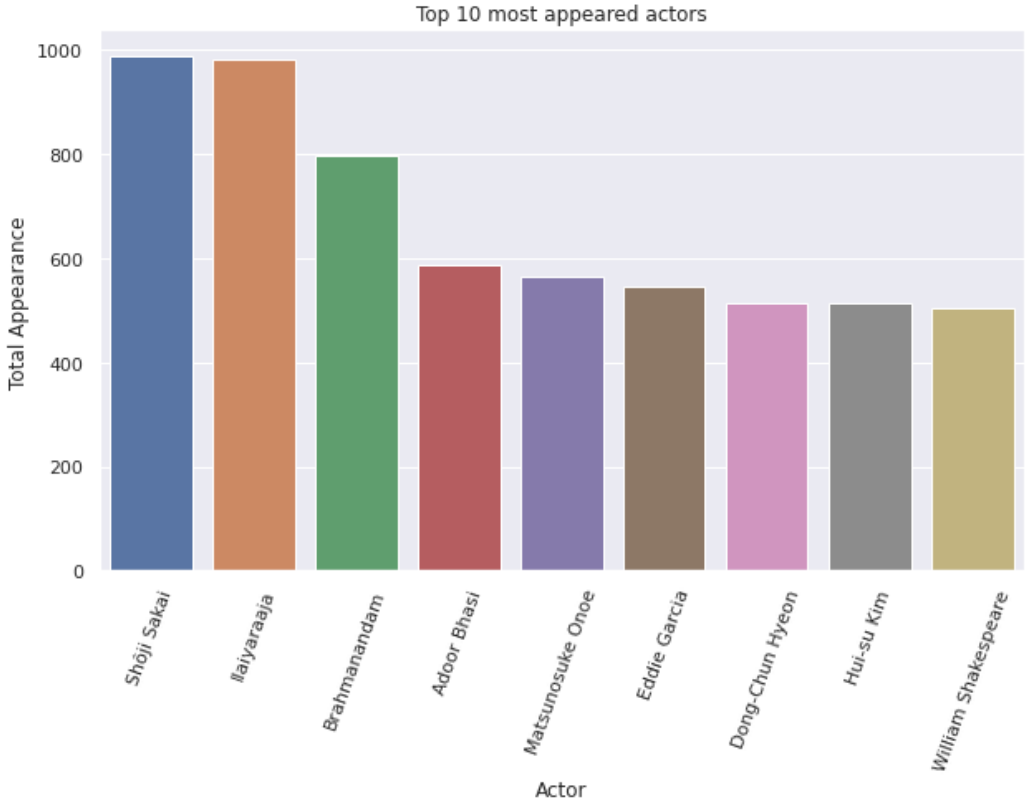


Figure 8: Most appeared actors

4 Finding Frequent Itemsets with the Apriori Algorithm

Items that occur either as singletons or together are considered as frequent if they appear more than some prespecified value. This value is called support. Support of an itemset is the percentage of the baskets which have those itemsets. In order to find the itemsets that occur in the dataset more than the support, scanning all possible combinations of the items is necessary, however this process takes long time if the dataset is huge. A-Priori algorithm helps reduce the runtime by considering only frequent items while generating possible combinations of items, namely candidates.In other words, according to the Apriori algorithm if an itemset is frequent, then all of its subsets are frequent.

Apriori algorithm starts from single itemsets and creates larger sets by combining sets that have been found frequent in the previous scan. Leaving

out the itemsets that are not frequent is how the Apriori algorithm reduces the total time of execution.

In order to implement the Apriori algorithm from scratch, first, functions to help building the algorithm have been written. Generating candidates, scanning and computing the supports of the itemsets and finding frequent itemsets with support greater than the defined minimum support. Algorithm has been run until there is no itemsets that meet the support criterion.

In order to find the frequent movie actors in IMDB dataset, first, support has been defined.Total number of baskets is 522,758. Different percentages of the total basket number have been employed as support values and supports that are greater than or equal to 0.2% of the total basket number have been found useless. No itemsets have been given by the algorithm as frequent itemsets with a support greater than that fraction. Additionally, using supports less than 0.04% has made the algorithm's computation time longer. Therefore, in this analysis, Apriori algorithm has been run with support values between the range 0.1% and 0.05% of the total basket number.

Runtime against different supports has been depicted in Figure 9.

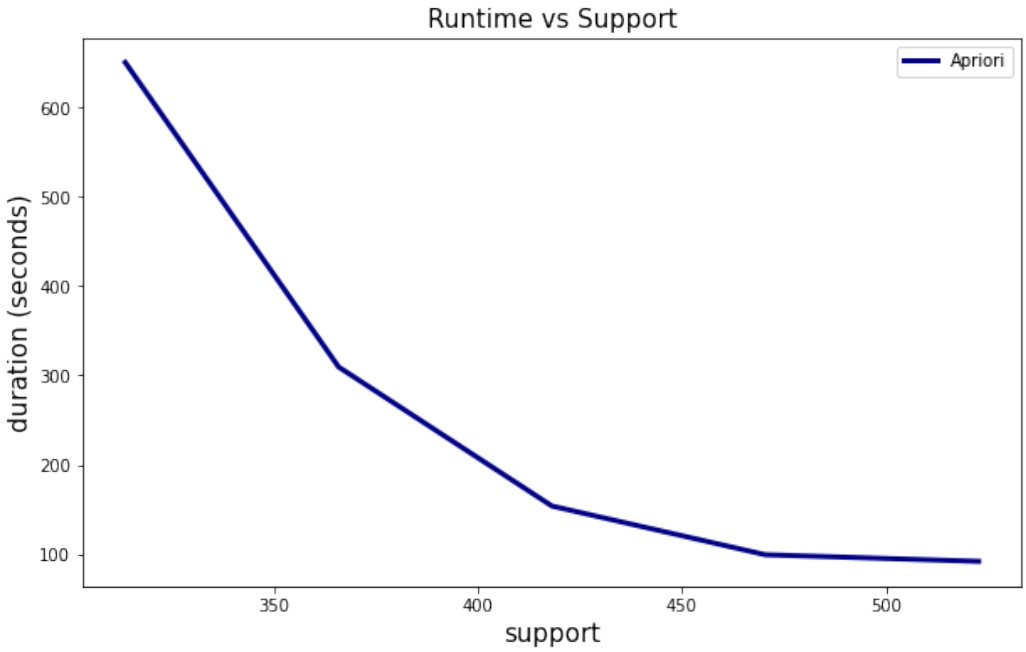


Figure 9: Apriori Algorithm Runtime vs. Support

5 Finding Frequent Itemsets with FP Growth

FP Growth algorithm is much faster than Apriori regarding the runtime, because it scans the data only twice while Apriori scans the data for every potential frequent item.FP Growth algorithm first builds the FP-tree and then scans the tree for frequent itemsets from that tree. This is the major advantage FG Growth has. In this analysis FP Growth algorithm has been implemented with FP-growth package of pyspark's Mllib library on the same dataset with the same support values. It can be seen that with FP-Growth algorithm computation time does not change drastically with the support value. The same itemsets as Apriori has been found frequent by FP Growth algorithm.

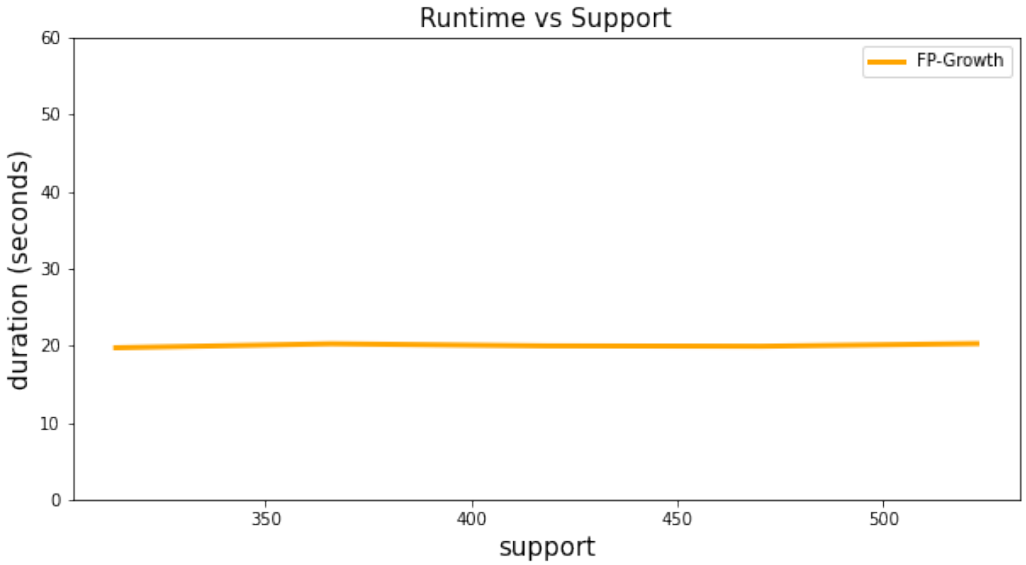


Figure 10: FP Growth Algorithm Runtime vs. Support

6 Apriori vs. FP Growth

Two algorithms have been both performed on the same transaction data and same itemsets have been given by them. However, with respect to computation time Apriori algorithm has been proved to be much slower than FP growth. Runtime in seconds against various supports has been plotted below. It is obvious that with lower support values Apriori algorithm requires much more computation time as opposed to FP-growth.

As for the scalability, algorithms have also been run on different samples taken from the basket data with the same support. Both algorithms have shown that computation time increases linearly with the sample size.

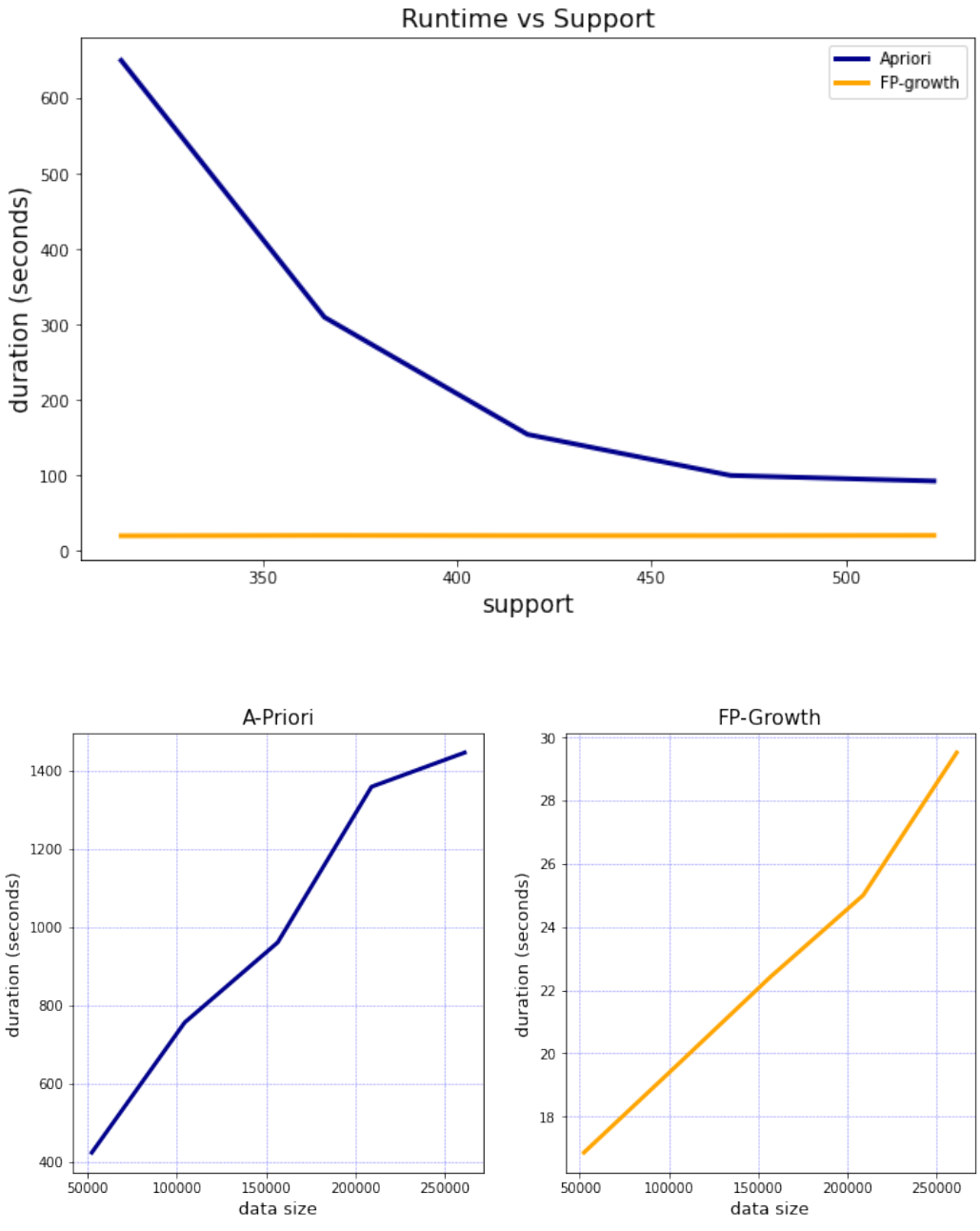


Figure 11: Runtime vs. Data Size

7 Conclusion

Apriori and FP growth algorithms both gave the same lists of itemsets as frequent; however FP growth algorithm was proved to be much faster than Apriori algorithm. Changing the support value, it has been seen that the computation time of Apriori algorithm increases exponentially whereas the execution time of FP growth stays almost the same.

When the data size has been decreased to compare the performances of these two algorithms on a much smaller dataset, FP growth still performs faster than Apriori. However, unlike changing supports, changing the size of the samples taken randomly from the basket data had the same linear effect on the execution times.

"I declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study."