

Application of Apriori Algorithm on the given MBA Dataset

Jehezkel Rosh T. Rodriguez
College of Computer Studies
Silliman University
jehezkelrodriguez@su.edu.ph

I. Abstract

Market Basket Analysis (MBA) serves as a fundamental tool for uncovering significant patterns within consumer purchasing behavior. In this study, The researcher used the Apriori algorithm to analyze the dataset, seeking to extract meaningful associations among items frequently bought together. The Apriori algorithm, known for its effectiveness in identifying frequent itemsets, is employed to reveal latent relationships and dependencies within the dataset.

The method in this study involves dataset preprocessing, application of the Apriori algorithm to uncover frequent itemsets, and the subsequent generation of association rules. By setting a minimum support threshold, noisy data are filtered out. The resulting association rules are evaluated based on confidence, providing a comprehensive understanding of product relationships.

The selected dataset, representing authentic retail transactions, forms the basis in the analysis. This research aims to contribute to the field of market basket analysis by showcasing the practical application of the Apriori algorithm and its ability to uncover valuable insights from transactional data.

The researcher then found out after the implementation of the algorithm, out of 14,964 transactions, there are 125 frequent itemsets and in which 19 generated association rules are derived with the minimum support of 0.005 and minimum confidence level of 0.1. This sheds a light to the purchasing habits of customers which then can be harnessed to optimize various components in business operations.

II. Introduction

The analysis of grocery data and association rule mining has become increasingly crucial in the world of retail and marketing. With the vast amounts of transactional data generated in grocery stores, understanding consumer purchasing patterns has emerged as a key strategy for businesses. Association rule mining, particularly through algorithms like Apriori, offers a powerful approach to

uncovering hidden relationships among products. By identifying frequently co-occurring items in transactions, businesses can gain insights into consumer behavior, optimize inventory management, and enhance the overall shopping experience.

Statement of the Problem

The researcher aims to address the following research question: **What are the association rules generated from the Apriori algorithm when applied to the given MBA datasets?**

III. Review of Related Literature

Market Basket Analysis has been widely researched and applied in various industries, particularly in retail. The foundational work by Agrawal, Imielinski, and Swami [1] introduced the Apriori algorithm, which has since become a cornerstone in association rule mining. The technique involves identifying frequent itemsets and generating association rules, providing valuable insights into customer purchasing patterns. Several studies [2] have explored variations of the Apriori algorithm and its applications in different domains.

The Apriori algorithm, proposed by Agrawal et al. [1], remains a key algorithm in association rule mining. The algorithm efficiently discovers frequent itemsets and generates association rules based on a priori knowledge. Numerous studies have extended and optimized the Apriori algorithm, such as the FP-Growth algorithm proposed by Han et al.[2], which addresses some of the limitations of the original Apriori algorithm.

Python has become a popular programming language for data science and machine learning due to its simplicity and versatility. The 'pandas' library is a fundamental tool for data manipulation and analysis. Wes McKinney's book [3], "Python for Data Analysis," provides a comprehensive guide to using 'pandas' for data-related tasks.

[4] The 'mlxtend' library is an extension of 'scikit-learn' and provides tools for exploratory data

analysis, feature engineering, and association rule mining. Raschka [5] provides insights into 'mlxtend' and its applications in machine learning in his book "Python Machine Learning."

IV. Methodology

Dataset

The [6] grocery dataset used in this study is provided to the researcher. The dataset can also be accessed through kaggle, providing transparency and reproducibility. The dataset contains the details member number, date purchased, item description, year, month, day and day of the week.

	A	B	C	D	E	F	G	H
1	Member	Date	itemDescr	year	month	day	day_of_week	
2	1808	#####	tropical fr	2015	7	21	1	
3	2552	5/1/2015	whole mil	2015	5	1	4	
4	2300	#####	pip fruit	2015	9	19	5	
5	1187	#####	other veg	2015	12	12	5	
6	3037	1/2/2015	whole mil	2015	1	2	4	
7	4941	#####	rolls/buns	2015	2	14	5	
8	4501	8/5/2015	other veg	2015	8	5	2	
9	3803	#####	pot plants	2015	12	23	2	
10	2762	#####	whole mil	2015	3	20	4	
11	4119	#####	tropical fr	2015	12	2	2	
12	1340	#####	citrus fruit	2015	2	24	1	
13	2193	#####	beef	2015	4	14	1	
14	1997	#####	frankfurte	2015	7	21	1	

Figure 1: A snippet of the uncleaned MBA dataset.

Prior to applying the Apriori algorithm, the dataset underwent preprocessing to ensure data quality and relevance. Steps included handling missing values, removing duplicates, and encoding categorical variables. Additionally, outliers were identified and addressed to mitigate their impact on the association rule mining process. The cleaned dataset was then formatted into a suitable structure for input into the Apriori algorithm.

0	1	2	3	4	5	6	7	8	9	10
whole mil pastry	salty snack									
sausage	whole mil semi-finish yogurt									
soda	pickled vegetables									
canned bc misc	beverages									
sausage	hygiene articles									
sausage	whole mil rolls/buns									
whole mil soda										
frankfurte soda	whipped/sour cream									
frankfurte curd										
beef	white bread									
butter	whole milk									
frozen veg other vegetables										
tropical fr sugar										
butter mil specialty chocolate										
frozen mc dental care										
rolls/buns rolls/buns										
root veg detergent										
sausage	rolls/buns									
dish clean cling film/bags										
canned bc frozen fish										
pip fruit	whole mil tropical fruit									
root veg whole mil pastry										
rolls/buns red/blush chocolate										

Figure 2: A snippet of the cleaned MBA dataset.

The Apriori algorithm was implemented using the 'mlxtend' library in Python. The algorithm follows a

two-step process: (1) identifying frequent itemsets, and (2) generating association rules. The 'mlxtend' implementation efficiently navigates the dataset, employing the Apriori principle to iteratively discover frequent itemsets of varying lengths. Subsequently, association rules were generated based on these frequent itemsets, considering the specified minimum support and confidence thresholds.

The choice of support and confidence thresholds is a critical aspect of association rule mining. In this study, the support threshold was set to 0.005, reflecting the minimum proportion of transactions that must contain a specific itemset for it to be considered frequent. A confidence threshold of 0.1 was chosen to ensure that only rules with a strong association between antecedent and consequent items were retained. These parameter settings were determined through an iterative process, balancing the need for meaningful associations with computational efficiency.

```

1 import pandas as pd
2 from mlxtend.preprocessing import TransactionEncoder
3 from mlxtend.frequent_patterns import apriori
4 from mlxtend.frequent_patterns import association_rules
5 import os
6
7 import os
8 for dirname, _, filenames in os.walk('/kaggle/input'):
9     for filename in filenames:
10         print(os.path.join(dirname, filename))
11
12 dataset = pd.read_csv('0./3mlxtend_files/William Academic 5h1z/3rd year/basket.csv')
13 dataset.fillna(0, inplace=True)
14
15 Transactions = []
16
17 for i in range(1493):
18     transaction = []
19     for j in range(11):
20         if dataset.iloc[i, j] != '':
21             transaction.append(dataset.iloc[i, j])
22     Transactions.append(transaction)
23
24 te = TransactionEncoder()
25 te_bin = te.fit_transform(Transactions)
26 Transactions = pd.DataFrame(te_bin, columns=te.columns_)
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure 3: A code snippet of the python program showing 'pandas' and 'mlxtend' being used in the algorithm.

V. Results

Frequent Itemsets

The application of the Apriori algorithm on the grocery dataset resulted in the discovery of 126 frequent itemsets. These itemsets represent combinations of products that are frequently purchased together by customers. The support values, indicating the proportion of transactions containing each itemset, provide insights into the popularity of these combinations.

	support	itemsets
0	0.021386	(UHT-milk)
1	0.008087	(baking powder)
2	0.033950	(beef)
3	0.021787	(berries)
4	0.016574	(beverages)
..
121	0.011629	(soda, whole milk)
122	0.005814	(soda, yogurt)
123	0.008220	(whole milk, tropical fruit)
124	0.005213	(yogurt, tropical fruit)
125	0.011161	(yogurt, whole milk)

Figure 4: Results in python showing the first and the last 5 frequent itemsets.

Association Rules

There are 19 association rules derived from the frequent itemsets that contribute to the understanding of customer behavior. Each rule consists of an antecedent and a consequent, along with confidence and support as its metrics. The association rules are as follows:

Rule	Support	Confidence
Frankfurter -> other vegetables	0.00515	14%
Bottled beer -> whole milk	0.00715	16%
Sausage -> whole milk	0.00896	15%
Newspaper -> whole milk	0.00561	14%
Domestic eggs -> whole milk	0.00528	14%
Frankfurter -> whole milk	0.00528	14%
Pork -> whole milk	0.00501	14%
Pip fruit -> whole milk	0.00662	13%
Citrus fruit -> whole milk	0.00715	13%
Shopping bags -> whole milk	0.00635	13%
Yogurt -> whole milk	0.01116	13%
Canned beer -> whole milk	0.00601	13%
roll/buns -> whole milk	0.01397	13%

Pastry -> whole milk	0.00648	13%
Other vegetables -> whole milk	0.01484	12%
Tropical fruit -> whole milk	0.00822	12%
Soda -> whole milk	0.01163	12%
Bottled water -> whole milk	0.00715	12%
Root vegetables -> whole milk	0.00755	11%

Table 1: Association rules with its support and confidence

```

1. Rule:'frankfurter' -> 'other vegetables'
Support: 0.00515
Confidence: 14%

2. Rule:'bottled beer' -> 'whole milk'
Support: 0.00715
Confidence: 16%

3. Rule:'sausage' -> 'whole milk'
Support: 0.00896
Confidence: 15%

4. Rule:'newspapers' -> 'whole milk'
Support: 0.00561
Confidence: 14%

5. Rule:'domestic eggs' -> 'whole milk'
Support: 0.00528
Confidence: 14%

6. Rule:'frankfurter' -> 'whole milk'
Support: 0.00528
Confidence: 14%

```

Figure 5: Results in python showing the first 6 out of 19 association rules derived from the frequent itemsets.

Analysis

The association rules derived from the analysis of the dataset reveal several patterns in customer purchasing behavior. Remarkable items like 'bottled beer,' 'sausage,' 'newspapers,' and 'domestic eggs' are frequently associated with the purchase of 'whole milk,' each exhibiting a confidence ranging from 14% to 16%. This indicates a consistent preference for combining these products in shopping transactions. Additionally, common grocery items such as 'rolls/buns,' 'pastry,' 'other vegetables,' and 'tropical fruit' also exhibit a significant association with 'whole milk,' ranging from 11% to 13%. This suggests that customers tend to purchase these items together.

Conclusion

The association rule analysis provides valuable information into the purchasing patterns of the customers in the dataset. The frequent associations between items like 'bottled beer,' 'sausage' and 'newspaper' with 'whole milk' shows a consistent consumer preference offering the retailers actionable information for product placement and targeted marketing. With the confidence of each rule consistently staying above the minimum threshold, this shows that the reliability of each rule is dependable. These findings contribute to the data-driven understanding of consumer behavior in the grocery retail area, emphasizing the importance of association rule mining.

IV. References

- [1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. *ACM SIGMOD Record* 22, 2 (1993), 207-216. DOI:<https://doi.org/10.1145/170036.170072>
- [2] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (2000). DOI:<https://doi.org/10.1145/342009.335372>
- [3] McKinney, W. (2017). Python for Data Analysis. *O'Reilly Media*.
- [4] Raschka, S. (2018). Python Machine Learning. *Packt Publishing*.
- [5] Raschka, S. (2021). mlxtend: Extensions and Helper Functions for Python's Data Analysis Library. *Journal of Open Source Software*.
- [6] Rashik Rahman. 2020. Groceries dataset for Market Basket Analysis(MBA). *kaggle*. Retrieved November 19, 2023 from <https://www.kaggle.com/datasets/rashikrahmanpritom/groceries-dataset-for-market-basket-analysismba/>