**Association Rules Item Moving Recommendations**

**Executive Summary**

Dillard’s retail store has collected an extremely large amount of sales data across their many locations. They are looking to optimize their store layout to increase sales, and this data can be a valuable resource in the best ways to rearrange items.

The data can be aggregated to evaluate what items were purchased together by each customer. The lists of customer ‘baskets’ can then be analyzed to look for patterns and associations between items purchased in the same ‘basket’. In such a manner, it is possible to discover items that, when purchased, ‘imply’ that another item will then be purchased with a higher probability. Though these relationships may not be causal, the associations discovered reveal valuable insight about consumer behaviors and potential methods for increasing sales.

Through this preliminary analysis, I have isolated 100 sku candidates for relocation within the store. For the initial move, I would recommend placing the correlated items together to maximize the impact of customer purchasing habits. Placing the items together is a safe method for capitalizing on customer habits because a customer buying the initial, ‘implying’ item, is significantly more likely to see and then purchase the implied item.

**Problem** **Statement**

Dillard’s would like to rearrange their store to increase store profits. They only have limited manpower to move 20 items, but they would first like an initial list of 100 skus in which a relocation has large potential to increase profits. They have provided a large sample of point of sales data to aid in the analysis for choosing the 100 skus

**Assumptions**

* The day that I evaluated is representative of all other days of sales for Dillards
* The stores that I sampled are representative of the smaller store sales
* Any item purchased on the day evaluated will not be returned on a later date
* Varying discounts did not influence sales is a manner that would give a false association signal

**Methodology**

The first step was to understand the data and the significance of each table and the columns within the tables. I will elaborate on the methods and difficulties with this in the analysis section. After correctly labelling the data, I began querying the transact database for customer transactions because this is the primary source of data to understand customer purchasing habits. Unfortunately, this dataset is enormous, so I needed to develop a technique to properly subsample the data in a manner that did not hurt the data integrity.

Since the dataset has approximately 20 million rows, any conditional statement or filter when querying data will be applying on every tuple and take an extremely long time. For this reason, I wanted to use the simple limit function in sql queries. There are a couple issiues with this method. The first concern is that by randomly sampling data, it is very likely that we will be evaluating incomplete customer baskets. This will prevent the discovery of meaningful relationships between purchases. Additionally, the limit function returns purchases sorted by sku, which will heavily skew the results. One possible workaround for the sku sorting would be sort by random() within sql, but this would generate a random number for each entry, causing an incredibly large and unnecessary number of operations. Additionally, this would not solve the first problem of incomplete baskets.

Instead, I decided to query a single day and save the results via pandas.to\_pickle() for fast reading at a later time. By querying a single day, I guarantee that each basket will be complete because it is impossible for a transaction to occur over multiple days. Querying a single day leaves the risk of seasonality or changing purchasing habits by day, so additional days should be tested as well to check results and protect against these risks.

After loading the data, it must now be cleaned. The first cleaning action is to remove all returns. Though it is important to track if an item has been returned later, the general inclusion of returns as transactions would provide false and harmful relationships. After filtering out returns, any column used for calculations must be converted to numeric figures because sql returned them as strings.

After basic cleaning of the data, we must narrow the dataset down even further to prevent RAM overload. To make sure that the filtering of the dataset keeps baskets intact, I decided to narrow down by store. Random stores would be a potential option if there was time to run multiple iterations, but this is unfortunately not an option. Instead, I selected the stores with the highest revenue since they have the largest impact on the chain’s finances and are the most important stores to focus on.

After selecting the maximum number of stores without crashing the kernel (40), I grouped all of the purchases made at these stores by their ‘basket’ and used the mlxtend package in python to mine association rules. I will elaborate on hyperparameterization in the analysis section

**Analysis**

The first step in analyzing the dataset was determining what each table and column within the tables meant. Luckily, there was a schema provided that explained the purpose and contents of each of the tables. Unfortunately, the columns were not correctly labelled in the trnsact table and there were two additional columns. It was a trivial matter to label most of the columns using the details provided in the schema, however there were a couple uncertain columns. C14 did not seem to align with any values given in the schema and has only 2 binary values, meaning it is likely an unnecessary variable. There were no sale prices included in the schema for trnsact table, but it appeared that these were the values in columns C10 and C11. Additionally, there was an sprice variable in the data description included on the second page of the schema document. A query across the whole dataset revealed only 7 values where c10!=c11, so these variables are functionally the same – both sale price. And finally, there was uncertainty on which of c12 and c5 was seq and which was inerid. Since seq is included in the key for the transaction table, it must be used to reduce the number of individual purchases in the process of bundling by overall transaction. When testing each of these columns as seq, only c5 as seq reduced the number of unique combinations, meaning c5 must be seq.

In order to choose the hyperparameters for the apriori and association rules functions, I needed to prioritize between support, confidence, and lift. Since I am not able to include any more data points without crashing my kernel, these are the only variables to adjust to reach the 100 suggest skus optimally. Since the number of data samples is quite small when compared to the number of unique items purchased (34415, 37203), the resulting supports were quite low. This means in order to get areasonable number of results, minsupport needed to be a relatively low value. Since minsupport is low, this can lead to extremely high and potentiall unreliable lift values. To combat this, I prioritized confidence as the primary metric, requiring a confidence of 60% to be included as a recommendation. I then lowered the minsupport until I was able to reach the necessary 100 skus.

The resulting table is below:

antecedants consequents support confidence lift

321 6072521 6032521 0.000581 0.600000 1474.928571

613 6072521 6032521 0.000581 0.600000 1720.750000

327 6072521 6062521 0.000581 0.800000 1529.555556

319 6062521 6032521 0.000523 0.666667 1638.809524

612 6062521 6032521 0.000523 0.666667 1911.944444

326 6062521 6072521 0.000523 0.888889 1529.555556

610 6062521 6032521 0.000465 0.750000 1843.660714

64 1949753 2419753 0.000436 0.600000 1290.562500

33 1389834 9230294 0.000436 0.733333 48.533974

567 3524026 3978011 0.000407 0.714286 241.001401

318 6032521 6062521 0.000407 0.857143 1638.809524

320 6032521 6072521 0.000407 0.857143 1474.928571

611 6032521 6062521 0.000407 0.857143 1843.660714

258 919823 4980033 0.000378 0.923077 599.390421

271 959823 4980033 0.000349 1.000000 649.339623

608 6032521 6072521 0.000349 1.000000 1720.750000

609 6032521 6062521 0.000349 1.000000 1911.944444

566 3524026 3898011 0.000349 0.833333 477.986111

342 6480353 6470353 0.000349 0.750000 2581.125000

45 1977926 157926 0.000320 0.636364 2190.045455

74 2127926 1977926 0.000320 0.636364 1990.950413

75 1977926 2127926 0.000320 0.636364 1990.950413

419 7758362 9230294 0.000320 0.636364 42.116259

218 3949830 9230294 0.000320 0.636364 42.116259

391 711116 9370294 0.000291 0.600000 47.468966

343 6470353 6480353 0.000291 0.900000 2581.125000

44 157926 1977926 0.000291 0.700000 2190.045455

465 939823 919823 0.000291 0.600000 1588.384615

288 6939904 5129905 0.000291 0.600000 2581.125000

266 939823 4980033 0.000291 0.900000 584.405660

595 939823 919823 0.000291 0.600000 1720.750000

397 7329260 9230294 0.000262 0.666667 44.121795

388 7232521 6972521 0.000262 0.777778 3345.902778

256 869823 4980033 0.000262 1.000000 649.339623

561 3908011 3998011 0.000262 0.888889 477.986111

560 3908011 2726578 0.000262 0.888889 650.874704

592 939823 919823 0.000262 0.666667 1764.871795

631 6490353 6480353 0.000232 0.625000 2389.930556

635 7222521 7232521 0.000232 0.625000 3072.767857

556 2716578 3908011 0.000232 0.625000 896.223958

554 3908011 3988011 0.000232 0.625000 614.553571

572 9230294 9370294 0.000232 0.625000 49.446839

574 3537981 9230294 0.000232 0.625000 41.364183

580 3690654 3968011 0.000232 0.625000 398.321759

347 6490353 6480353 0.000232 0.875000 2509.427083

16 1149823 4980033 0.000232 0.875000 568.172170

239 4142521 4512521 0.000232 0.625000 2688.671875

289 5129905 6939904 0.000232 0.750000 2581.125000

345 6490353 6470353 0.000232 0.625000 2150.937500

238 4512521 4142521 0.000232 0.625000 2688.671875

637 6972521 7222521 0.000232 0.625000 4301.875000

394 7222521 7232521 0.000232 0.625000 2389.930556

386 7222521 6972521 0.000232 0.625000 2688.671875

387 6972521 7222521 0.000232 0.625000 2688.671875

389 6972521 7232521 0.000232 0.875000 3345.902778

149 4559772 3179772 0.000203 0.714286 4916.428571

84 4559772 1999772 0.000203 0.857143 4214.081633

274 979823 4980033 0.000203 0.857143 556.576819

553 1999772 3179772 0.000203 0.714286 4916.428571

85 1999772 4559772 0.000203 0.857143 4214.081633

83 1999772 3179772 0.000203 0.714286 4916.428571

597 9230294 4980033 0.000203 1.000000 649.339623

80 1988302 2478302 0.000203 0.857143 4916.428571

28 1240638 9230294 0.000203 0.714286 47.273352

398 7373522 9370294 0.000203 0.714286 56.510673

623 6320353 6300353 0.000203 0.714286 4916.428571

627 6480353 6470353 0.000203 0.714286 2458.214286

8 1089823 4980033 0.000203 1.000000 649.339623

634 7232521 7222521 0.000203 0.714286 3072.767857

552 4559772 3179772 0.000203 0.714286 4916.428571

334 6320353 6300353 0.000203 0.714286 4097.023810

338 6320353 6340353 0.000203 0.714286 4097.023810

60 2209753 1879753 0.000174 0.666667 2085.757576

335 6300353 6320353 0.000174 0.833333 4097.023810

293 5329905 7039904 0.000174 0.833333 2867.916667

81 2478302 1988302 0.000174 1.000000 4916.428571

324 6042521 6072521 0.000174 0.833333 1433.958333

545 2209753 1879753 0.000174 0.666667 3823.888889

591 919823 4980033 0.000174 1.000000 649.339623

86 208065 68065 0.000174 0.666667 3823.888889

87 68065 208065 0.000174 0.666667 3823.888889

59 1837926 2987926 0.000174 0.666667 3277.619048

336 6300353 6340353 0.000174 0.833333 4779.861111

56 1837926 1977926 0.000174 0.666667 2085.757576

101 2637662 4640325 0.000174 0.666667 4588.666667

53 1617926 2127926 0.000174 0.666667 2085.757576

50 1617926 1977926 0.000174 0.666667 2085.757576

43 1837926 157926 0.000174 0.666667 2294.333333

337 6340353 6300353 0.000174 0.833333 4779.861111

618 6042521 6062521 0.000174 0.666667 1433.958333

624 6300353 6320353 0.000174 0.833333 5735.833333

625 6340353 6320353 0.000174 0.833333 5735.833333

14 1308996 1138996 0.000174 1.000000 1376.600000

517 9600684 9469364 0.000174 0.666667 2294.333333

12 1119823 4980033 0.000174 0.833333 541.116352

339 6340353 6320353 0.000174 0.833333 4097.023810

4 1039820 2259942 0.000174 0.833333 3186.574074

90 2209753 2419753 0.000174 0.833333 1792.447917

184 3649770 9230294 0.000174 0.666667 44.121795

245 5871857 4441857 0.000174 0.666667 3277.619048

236 4111857 5871857 0.000174 0.666667 3823.888889

550 4559772 3179772 0.000174 0.833333 5735.833333

555 3908011 2716578 0.000174 0.833333 819.404762

295 5358963 9230294 0.000174 0.666667 44.121795

242 4421180 5917506 0.000174 0.666667 5735.833333

286 5129105 5329105 0.000174 0.666667 5735.833333

578 3968011 3898011 0.000174 0.833333 477.986111

273 969823 4980033 0.000174 1.000000 649.339623

544 1879753 2209753 0.000174 0.666667 3823.888889

323 6042521 6062521 0.000174 0.666667 1274.629630

234 4111857 4441857 0.000174 0.666667 3277.619048

237 5871857 4111857 0.000174 0.666667 3823.888889

262 929823 4980033 0.000145 0.800000 519.471698

309 5566264 6938708 0.000145 0.800000 2294.333333

38 1479911 939906 0.000145 0.800000 6883.000000

628 6470353 6480353 0.000145 1.000000 2867.916667

551 3179772 4559772 0.000145 1.000000 5735.833333

276 989823 4980033 0.000145 1.000000 649.339623

543 2209753 1879753 0.000145 0.800000 2502.909091

616 6042521 6062521 0.000145 0.800000 1529.555556

620 6320353 6340353 0.000145 1.000000 5735.833333

37 1479911 3058015 0.000145 0.800000 6883.000000

549 3179772 4559772 0.000145 1.000000 4916.428571

622 6300353 6320353 0.000145 1.000000 4916.428571

148 3179772 4559772 0.000145 1.000000 4916.428571

100 4640325 2637662 0.000145 0.800000 4588.666667

584 919823 4980033 0.000145 0.800000 519.471698

548 3179772 1999772 0.000145 1.000000 4916.428571

536 2127926 1977926 0.000145 0.800000 2502.909091

633 7222521 7232521 0.000145 1.000000 3823.888889

632 7222521 6972521 0.000145 1.000000 4301.875000

534 1479911 3058015 0.000145 0.800000 6883.000000

82 3179772 1999772 0.000145 1.000000 4916.428571

421 7762152 8072152 0.000145 0.800000 6883.000000

621 6320353 6300353 0.000145 1.000000 5735.833333

376 687171 797171 0.000116 1.000000 8603.750000

542 2209753 2419753 0.000116 1.000000 2150.937500

377 797171 687171 0.000116 1.000000 8603.750000

3 1009823 4980033 0.000116 1.000000 649.339623

518 1089823 4980033 0.000116 1.000000 649.339623

532 1479911 3058015 0.000116 1.000000 8603.750000

420 8072152 7762152 0.000116 1.000000 6883.000000

287 5329105 5129105 0.000116 1.000000 5735.833333

614 6062521 6072521 0.000116 1.000000 1720.750000

36 3058015 1479911 0.000116 1.000000 6883.000000

142 3058015 939906 0.000116 1.000000 8603.750000

39 939906 1479911 0.000116 1.000000 6883.000000

524 9230294 4980033 0.000116 1.000000 649.339623

243 5917506 4421180 0.000116 1.000000 5735.833333

535 939906 3058015 0.000116 1.000000 8603.750000

533 3058015 1479911 0.000116 1.000000 8603.750000

531 3058015 1479911 0.000116 1.000000 6883.000000

143 939906 3058015 0.000116 1.000000 8603.750000

92 2470168 2250168 0.000116 1.000000 8603.750000

93 2250168 2470168 0.000116 1.000000 8603.750000

530 3058015 939906 0.000116 1.000000 8603.750000

In addition to evaluating associations strictly based on support, confidence, and lift, it is also important to note that some of the antecedants are repeated, meaning that sku is of particular importance. The following chart lists the skus that appear most frequently to imply the purchase of another item:

6320353 5

6062521 5

3179772 5

7222521 5

3058015 5

6032521 5

2209753 5

4559772 4

6300353 4

6042521 4

3908011 4

939823 4

1479911 4

919823 3

6340353 3

1999772 3

1837926 3

9230294 3

939906 3

6972521 3

6072521 3

6490353 3

**Conclusions**

Since there was a severe limitation on processing power, the statistical confidence on these results is not ideal for a chain-wide decision. That being said, this analysis has shown very promising results to be confirmed with further analysis. There are a large number of items that have strong purchasing relationships with one another, leading to purchases at significantly higher rates than would likely be seen otherwise.

If Dillard’s were to move items without further analysis, I would recommend focusing on the items with the highest support and the items that appear the largest number of times as an antecedent. Focusing on the relationships with the highest support means the relationship discovered is much more likely to be statistically significant and to hold in the real world. Additionally, since there is a limit on moving 20 items, moving the items that imply multiple purchases may allow more additional purchases per move.

**Next Steps**

As mentioned earlier, the most significant next step is to evaluate the association rules with additional data. This could be done in a number of ways. With more time, a similar analysis could be run on different days across the year to mitigate the risk that my chosen day was an anomaly. With access to more computing power, the need to break up the dataset would be lessened or eliminated and larger amount of data could be analyzed.

Additionally, there are ways to modify the sku selection process to maximize profit further. This could be done by evaluating the expected value that a product move may have on profit. Additionally, such a method might also include custom minimum support values for higher cost, higher profit items that are sold less frequently.