Predictable Purchases

Jordan Deagan
Computer Science
New College of Florida
Sarasota, US
Jordan.deagan17@ncf.edu

Abstract—I wanted to see if I could predict the purchase of a single type of product given some of the habits of users on an online grocery store. I split the data I got into many data tables to train and test on. I then ran ID3(a decision tree algorithm) and Markov Chain(a state probability model) on these new tables. I recorded the results of both algorithms and created displays for the Markov Chains. The Markov Chains performed better than ID3, but there is much room for improvement overall. If dealt with again, more probability models and products will be tested to refine the results

I. INTRODUCTION

A. *Question*

In the area of purchasing behavior analysis, the most common query is what is the next purchased item of a customer based on the purchases of similar customers. I was curious if, instead of looking at the next item in a group, could I determine if a specific item would be purchased next. When given the time and day a person makes a purchase, as well as that person's previous purchases, can it be predicted that a specific item, or type of item, will be in their purchase. In answering this question, I will use an ID3 Decision Tree predictor and a Markov Chain probability model trained on all users in my data, and ones trained on only users who have previously bought that item, to predict if an item will be in the most recent purchase of a user.

B. *Reasoning*

I am asking this question because I want to find if human behavior is predictable when it comes to buying specific groceries. This question and its results are important to online stores as part of predictive software for advertising and recommendations, as well as knowing how much of a product to order, especially for perishable groceries.

I am using ID3 to see how precise a decision tree can get with under sampled data, and to see what rules are created when looking at human behavior. I am using Markov Chains because they are a state-based probability model that does not rely on the past. This means a user who hasn't bought an item before still has the probability to buy it in the future.

C. *Overview*

Once I found my data, I ran preprocessing to shorten and reorganize the data to better fit my algorithms. In the next section, I will describe the data I used and how I altered it. In Methodology I explain what ID3 and Markov Chains are, as well as how I use them. In Results, I have the tables and detailed results of running the algorithms. In Conclusions, I analyze the results to determine if my question was answered, and how my results could be used. In Future Works, I bring up potential alternative methods I could have used, and ways I could improve on my results or further tests I could perform.

II. DATA

The data I am using comes from Instacart's open source purchases information. The data originally came as 6 data tables. During preprocessing, I created several additional tables and files.

A. *Data Tables*

1) *Store Information*

The first data tables are on the departments, aisles, and products in the store. The department data table contains the department titles and IDs. The aisle data table has the same thing for every aisle. The products

data table provides the ID and name of every product, but also includes the aisle and department the product is found in. These tables were used in preprocessing to develop the lists of products from the appropriate locations. For example, Beef Jerky Dog Treats is not a product wanted in the list of Beef Jerky products, and it is not on the same aisle as other actual beef jerky products, so I do not include that aisle when grabbing beef jerky IDs.

The lists I created were based on searching for products names with the passed phrase in it, and filtering those results by the aisle the product was in. I would add the correct product's product ID to the list, saving those for later. The 3 created lists contained the product IDs of all appropriate "Beef Jerky", "Trail Mix", and "Protein Bars" products respectively.

2) *User Order*

The initial table contained all user purchases organized by user ID and purchase order. The table contained purchase IDs, the ID of the user who purchased it, if the purchase is in the prior, test, or training purchase data table, the number pertaining to that purchases location in order history of that user, the day of the week the purchase was made, the time of day it was made, and, if the purchase was not the first purchase of a user, the amount of days since the last purchase by this user is recorded. An excerpt of the table is shown below.

USER ORDERS

order id	*user id*	*eval set*	*order number*	*order day of week*	*order hour of day*	*days since prior*
2539329	1	prior	1	2	8	
2398795	1	prior	2	3	7	15
473747	1	prior	3	3	12	21
2254736	1	prior	4	4	7	29

Fig. 1: sample of user order data table

I decided to split this table into a training and a testing table, removing the testing data from the training process. I did this twice, as decision trees only need the most recent information to be put through the tree, but a Markov chain predicts the final purchase based on the previous purchase, thus the Markov Chain training set would be the same, but the testing set would have the two most recent purchases. To separate the data for training and testing, I split the purchases into the most recent purchases of each user and all prior purchases. Due to lack of access to the test purchase data table referenced in the eval set column, and those purchases always being the most recent purchase of that user, I would look ahead while splitting and add the current purchase to my test table if the next purchase was labeled test. For all other users, the purchase labeled train was the most recent purchase, and thus was used for pulling the appropriate purchase away. This created 3 new tables, one used in training both of my algorithms, and a testing table for each algorithm.

In addition to splitting the table, I altered the new tables once they were separate by replacing the unneeded eval set column with a new purchased item column, containing "true" if a specific purchase contains an item from the product list I give the table, and "false" if it does not. I also shortened the 'hour of day' column by grouping every 2 hours into one value. For instance, a purchase made at hour 3 is part of the 2-3 hour group, while a purchase made at hour 4 is part of the 4-5 hour group. I created three versions of each of my previous 3 tables, one for each product, with the training table creating another table containing only the purchases of users who have at least one purchase of an item listed for easier access.

3) *Order purchases*

The last two tables that came with the data were of the products in each order. The tables are for the purchases identified as prior and train in the eval set column in the previous table. They record the purchase ID, a product ID of a product in the purchase, the order in which that product was added to the purchase, and if that product had been purchased by the user before.

ORDER PRODUCTS

order id	*product id*	*add to cart order*	*reordered*
1	49302	1	1
1	11109	2	1
1	10246	3	0
1	49683	4	0
1	43633	5	1
1	13176	6	0
1	47209	7	0
1	22035	8	1

Fig. 2: sample of order product data table

With the previous reorganized split in the user order table, I also had to reorganize these tables as many purchases previously in prior were now in train, so I reorganized the tables accordingly. I then created 2 dictionaries of products, with the keys being order IDs, in order to have quicker access to each order and get rid of all the unnecessary information.

III. METHODOLOGY

A. Background

1) ID3

ID3 is a decision tree predictive algorithm. It works by looking at every attribute in a data table and branching the tree it creates based on which attribute can simplify the output the most. This process looks at what the branches would look like when split on individual values in the various attributes and determining the best attribute as which one splits the target value the most. As an example, ID3 is used to determine whether to play ball based on the weather conditions in the following data table

PLAY BALL WEATHER

Day	*Outlook*	*Temp*	*Humidity*	*Wind*	*Play Ball*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Fig. 3. Play Ball Example

The algorithm looks at the attributes 'Outlook', 'Temp', 'Humidity', and 'Wind', and determines that 'Outlook' best splits the values in 'Play Ball'. After that, branches are created based on the values in 'Outlook', with the branch based on 'Overcast' having only 'yes' in the 'Play Ball' column, and the other two branches being mixed. Those two branches are split again until they are pure. They create the following tree

Fig. 4. Play Ball ID3 tree

2) Markov Chain

Markov Chains are used to create probability models for a series of data points where only the current state of something is the only thing used for predicting the next state. The weights of the model, or the probabilities used, are created based on previous state transitions. As an example, there is a chain of events with states E and A. In the past, whenever the current state is E, 3 times out of 10, the next state is E again, while the other 7 times the next state is A. If the current state was A, 6 times out of 10, the next state was A again, with the other 4 times the state being E. This creates transition rates from each state to each state, so that when there are new events in the chain, the next state can be predicted based purely on the current state. If a new event is added and the current state is E, the probability that the next state is E as well is 30%. That is the probability regardless if the previous state is A or E.

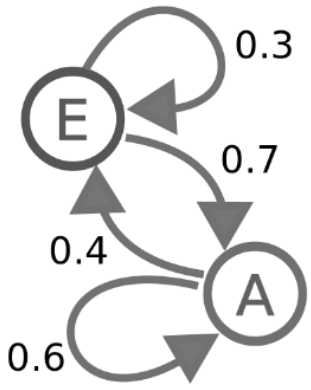

Fig. 5: sample Markov Chain

B. Process

1) ID3

For ID3 decision tree, I used the algorithm that I created in Project 3, with some slight altering to accommodate the different data. I ran the user orders training set with the focus value being the purchased item column. I excluded the order ID and the user ID from the attributes to split on, as they create unique identifiers that are not reliable. If a branch ran out of attributes to split on and was not pure yet, the branch returns the probability of purchasing the product based on the purchases left in the branch, while those that reach a pure set after splitting returns a definitive yes or no.

I ran ID3 on both the full user sets and the purchasing user sets of all three products, as to create 2 decision trees for each product to test. I save the decisions as a dictionary of dictionaries in a json file. I then test both decision trees on the training data and the testing data and record the results.

2) Markov Chain

To create the weights for the Markov Chain, I divided the number of specific transitions by the amount of times the first state occurs, i.e. the transition rates for purchasing a product to purchasing a product is created by the number of times that transition has happened in the past divided by the number of times purchasing was the current state during transitions. I did this on both training sets in each product, creating two lists for each product, with each list containing a list for each initial state, i.e. the list of transition rates for the full user set of Beef Jerky purchases is [[0.40,0.60],[0.00,1.00]], with the first interior list having the transition rates for the initial state of purchasing with the next state being purchasing and not purchasing respectively.

To test the transition rates, I put each purchase that wasn't the last purchase of a user into a program that randomly generated the next purchases result based on the transition rates of the state of the current purchase. I test both trained transition rates on both the training data and the altered testing data that contains the two most recent purchases. I repeat this process ten times and average the results in order to best represent the model.

IV. RESULTS

After running the tests on each algorithm, I recorded the true and false positives and negatives for the training and testing results, as well as calculating the accuracy of that test. I recorded the results in the tables below. S1 refers to the models trained on the entire user data table. S2 refers to the models trained on the data table containing only users who purchased the specific product before.

A trend among the decision tree tests is that the decision tree was so under sampled that it never predicted a purchase by anyone. This resulted in a highly accurate model, as the majority of the purchases do not contain the products, but it is not precise as it does nothing to determine when someone might purchase an item. The actual trees become very wide very quickly, and the differently trained versions within product tests, while branching different, still amount to the same results.

BEEF JERKY DECISION TREE

	Train		Test	
	Buy	Not Buy	Buy	Not Buy
Buy S1	0	0	0	0
Not Buy S1	11223	3128651	726	205483
Buy S2	0	0	0	0
Not Buy S2	11223	3128651	726	205483

Fig. 6. Predictions of Beef Jerky trained Decision Tree

	Train	Test
S1 accuracy	0.9964256527	0.9964256527
S2 accuracy	0.9964793001	0.9964793001

Fig. 7: Accuracy of Beef Jerky trained Decision Tree

TRAIL MIX DECISION TREE

	Train		Test	
	Buy	Not Buy	Buy	Not Buy
Buy S1	0	0	0	0
Not Buy S1	18673	3121201	1220	204989
Buy S2	0	0	0	0
Not Buy S2	18673	3121201	1220	204989

Fig. 8. Predictions of Trail Mix trained Decision Tree

	Train	Test
S1 accuracy	0.9940529461	0.9940836724
S2 accuracy	0.9940529461	0.9940836724

Fig. 9: Accuracy of Trail Mix trained Decision Tree

PROTEIN BAR DECISION TREE

	Train		Test	
	Buy	Not Buy	Buy	Not Buy
Buy S1	0	0	0	0
Not Buy S1	36117	3103757	2547	203662
Buy S2	0	0	0	0
Not Buy S2	36117	3103757	2547	203662

Fig. 10. Predictions of Protein Bar trained Decision Tree

	Train	Test
S1 accuracy	0.9884973091	0.9876484538
S2 accuracy	0.9884973091	0.9876484538

Fig. 11: Accuracy of Protein Bar trained Decision Tree

A trend among the Markov Chains is that the models trained on the smaller user sets were less accurate, but where more likely to predict that someone would buy an item. The rate of transition from purchasing to purchasing never changes, but it was more likely for someone to purchase an item after not purchasing it, usually because the full set transition rates were close to 0.00 and 1.00 for negative to positive and negative to negative respectively, while the smaller set predictions were closer to 0.10 and 0.90 instead. This causes an 8-9 percent difference in accuracy, an increase in true positives, but greatly increased number of false positives.

With each product, I include the transition rates for each model created to show the difference between the trained versions, and how close, overall, the models were to the each other for each product.

BEEF JERKY MARKOV CHAIN

	Train		Test	
	Buy	Not Buy	Buy	Not Buy
Buy S1	1674	2518	117	156
Not Buy S1	8890	2920583	609	205327
Buy S2	2250	265428	153	18568
Not Buy S2	8314	2657673	574	186915

Fig. 12. Predictions of Beef Jerky trained Markov Chain

	Train	Test
S1 accuracy	0.9961115533	0.9962906566
S2 accuracy	0.9066892096	0.9071733048

Fig. 13. Accuracy of Beef Jerky trained Markov Chain

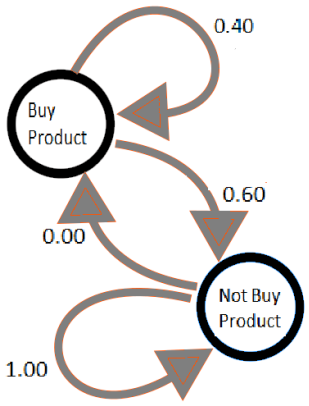

Fig. 14. Weights of Markov Chain trained on all users

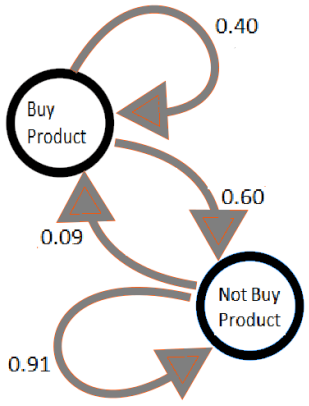

Fig. 15. Weights of Markov Chain trained on purchasing users

TRAIL MIX MARKOV CHAIN

	Train		Test	
	Buy	Not Buy	Buy	Not Buy
Buy S1	2521	4103	186	275
Not Buy S1	15156	2911885	1035	204714
Buy S2	3620	294501	251	20726
Not Buy S2	14057	2621487	969	184263

Fig. 16. Predictions of Trail Mix trained Markov Chain

	Train	Test
S1 accuracy	0.9934351400	0.9936481919
S2 accuracy	0.8948216310	0.8947912070

Fig. 17. Accuracy of Trail Mix trained Markov Chain

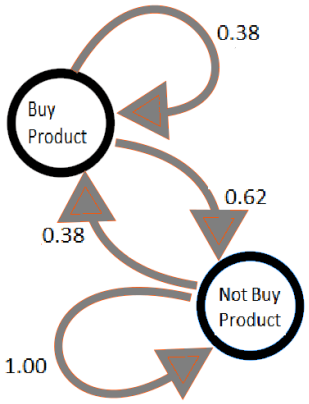

Fig. 18. Weights of Markov Chain trained on all users

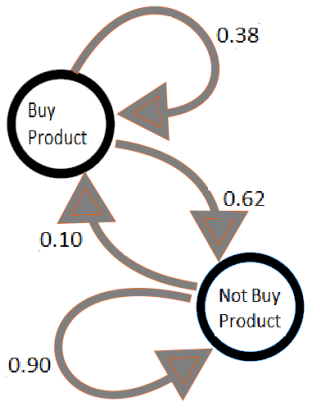

Fig. 19. Weights of Markov Chain trained on purchasing users

PROTEIN BAR MARKOV CHAIN

	Train		Test	
	Buy	Not Buy	Buy	Not Buy
Buy S1	5365	36728	380	2572
Not Buy S1	28627	2862945	2167	201090
Buy S2	6800	238385	495	16773
Not Buy S2	27192	2661288	2052	186889

Fig. 20. Predictions of Protein Bar trained Markov Chain

	Train	Test
S1 accuracy	0.9777223030	0.9770194317
S2 accuracy	0.9094727244	0.9087057306

Fig. 21. Accuracy of Protein Bar trained Markov Chain

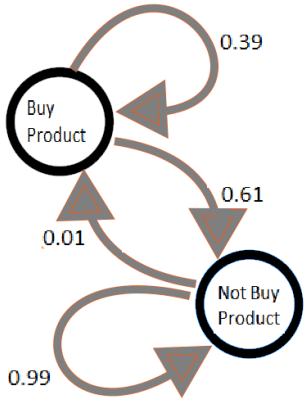

Fig. 22. Weights of Markov Chain trained on all users

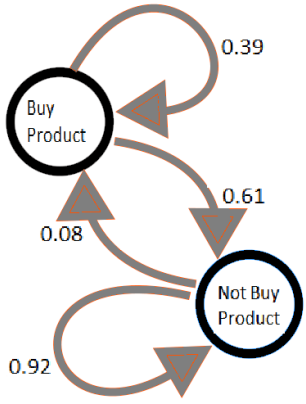

Fig. 23. Weights of Markov Chain trained on purchasing users

V. CONCLUSIONS

Overall, I believe that, with the correct algorithm and training, this process can e useful for identifying how much a perishable product might need to be purchased. While the ID3 algorithm did not perform well, showing that simple decision trees are not as well equipped to deal with severely under sampled data with a small range of attributes to look through, i.e. 4 split capable attributes with a ratio of positives to negatives ranging from 1:86 to 1:279. On the contrary, Markov Chains did decently, usually getting between 15-20% of the positive results correct, while getting between 99.9% and 90% of the negative results right. With a different algorithm, or with better tuning and training, a state based probability model could get produce results that are more precise, with less false results, or could get the mount of false positives to be close to the amount of false negatives, thus providing an accurate representation of the products future demand and thus how much of that product a store would need to purchase during restocks.

VI. FUTURE WORKS

If I come back to this problem in the future, I will compare Markov Chains against other similar models, as well as testing the models on a wider variety of products, from perishables and frozen foods to toiletries and cleaning supplies to see if these models work on all products in a grocery store.

REFERENCES

[1] "The Instacart Online Grocery Shopping Dataset 2017", Accessed from https://www.instacart.com/datasets/grocery-shopping-2017 on 10/26/18