Throughout this course I learned a lot about data mining. I came in with no knowledge about any data mining techniques and I am now leaving having a great baseline of what data mining truly is about. One technique that we learned that I was fascinated about was the Apriori algorithm. I enjoyed learning how to compute the frequency of items, support, confidence, and lift. One of the most common applications of Apriori was using this algorithm to use it in a market basket analysis on grocery data. I was able to find a great dataset provided by Kaggle that allowed to me use its data and the Apriori algorithm. In this project there are four major questions I had asked myself and kept in mind as I was working through the project. First question was, how do I want to sort my data? I was given a lot of information such as customer ID’s, item names, and purchase data. I was continually asking myself, what data matters to me and what data is irrelevant. The second question I asked myself was, how many combinations of products do I want to look for? Do I want to look for just pairs of items bough together, triples, groups of 4, and so on? This question went hand in hand with my third question which I asked myself, what is a good support to hold for my data. If I chose a support threshold too high my data would be too sparse, and it wouldn’t give very much information to look at. On the other hand, if I kept my support threshold too low, I would have an overflow of data, and it would make computation significantly slower. This question I struggled with a lot since I didn’t want to sort out too much data but also didn’t want to keep irrelevant data. That question then leads me onto my fourth question, how can I keep my code efficient and scalable. With those four questions in mind, I think it helped me stay on track on what I wanted the output of my project to look like. There were two notable challenges that I had to overcome in this project. The first being what to do with my data. My dataset contained 38,765 rows of data. Each row contained data about the customer, item purchased, and purchase date. I had to spend a lot of time sorting through the data. I found many ways on how to organize it, and I ended up keeping two sets of sorted data that I used in my algorithm. My second major challenge was optimizing my code. This was the real struggle for me. I was able to initially figure out a solution and get expected results with a small sample. But as I began using the dataset, I realized that just for finding pairs of groceries my computation time was very long; and I knew it would only get exponentially longer when trying to compute triple items and so on. My end results ended up being very informational. I was able to see all the products bought in groups of two, three, and four; and was able to see the confidence scores for groups of, two, three, and four. I will elaborate in greater detail the process of my work in the upcoming sections.

I tasked myself to find the frequently bought together pairs, triples, and groups of four grocery items. Once I had computed the frequency of each, I would then compute the confidence scores of each. The input data I was given was a csv file containing 38,765 rows of data. Each row of data contained a member number, date of purchase, and item description. The output of my data is a dictionary containing keys which are the groups of two, three, and fours and the values of each key are the number of times each group occurred. My second output data is like my first. The keys will be the same as in it’ll be groups of two, three, and four; but the values will be the confidence scores. An example of finding a group of triples would be…



This kind of format is what the expected output looks like. Each triple will contain a frequency and confidence value. My first question I was asking myself was how I want to sort my data. As mentioned for each row in my input data I was given a member number, date of purchase, and item description. When looking through different datasets, I always kept in mind what were the main columns of data I really needed. I had decided on the one I have chosen since it had exactly what I needed. I did not end up using the purchase date column in my algorithm. In my eyes it was not a very relevant piece of information that I needed to keep in my final output. I decided I would need to keep the member number and item description, as those two were the most important pieces of data. My second question I had in mind was how many different combinations I want to compute. This question was one of my most important questions I kept asking myself throughout the project. I could’ve easily just done groups of two and leave it at that. Doing so would’ve left me with a lot of unused data, as in it would’ve felt incomplete. I also didn’t want to compute too many groups of items for two main reasons. The first one being eventually I would be left with little to no data to filter through, it would be too sparse and provide little to no information that I am interested in. I say that, because let’s say a customer buys six items in their cart, but it only happens three times. That isn’t a large enough sample for it to be frequent enough to be considered. My second reasoning was for computational time. As I look for more groups of items my computation time increased while my returned results decreased. At some point the payoff was not worth it in the amount of time to compute versus the results returned. My third question I asked myself was, what do I want my support to be set to. Much like my second question, I needed to find a number that would exclude just enough information where I was able to get a healthy returned amount of data from each iteration. Finding the correct support was difficult, but once I was set on a number, it made my output very precise and understandable. My last question I kept in mind that tied into a lot of my other questions is, how do I keep my code efficient. I had a lot of scalability issues, and it took a lot of thinking, testing, and rewriting to eventually find an acceptable solution that was efficient. This last question was also of my main challenges working on this course project. Writing scalable code was my primary key challenge. My second main challenge was figuring out how to organize the data. I spent a large part of my time on the project working out those two challenges.

My algorithm can be explained in three different parts. There is the preprocessing, frequency, and confidence computing sections. The preprocessing section included three separate parts. The first part was reading the data using pandas read.csv(). Once I had that data saved into a variable, I went into my second part of preprocessing. I had a function called occurrences(), where I was sorting my data by the ‘itemDescription’ which in simpler terms is the name of each grocery item. Once I had that sorted, I would loop through each row and compute how many times each item was bough and save it into a dictionary. That function would return a dictionary containing unique keys for each item and an integer value containing the occurrence of each item. My third step of preprocess was computing each customer’s cart. Originally the data looked like this…

The goal was to transform it into this…



This is a very simplified version of the data, but the goal was to return a dictionary that contain a unique number of keys that represented the member number and a list of values containing the items. This preprocessing step answered my first question on how to sort the data. I spent a lot of time trying to figure out what exactly I needed. I was stuck in a box thinking I only needed one dictionary containing all the data, but it ended up being more intuitive to create two separated dictionaries to store data. So, the output of my preprocessing step includes a dictionary of items and their frequency while the other dictionary contains member numbers and all the items they have bought. My second portion of the project included finding the frequency of pairs of items bought together. This is where I had to use my preprocess data to find the correct output. I would begin by making a dictionary that included keys of each possible group of items. I would then go through the other dictionary of each customer and their items bought and count how many times this group occurred. I did this for groups two, three and four. Two of my main questions of figuring out how many combinations to compute and finding the right support were figured out in this part. I figured using a support of 18, gave me just enough information for each grouping to have valuable information. With that information I was able to find an adequate number of twos, threes, and groups of four bought together. The main hiccup I had was having efficient code. My computation time was originally upward of five minutes. I knew this was too long and spent a lot of time on the whiteboard figuring out how to bring down the run time. Finding the number of pairs was not an issue, that ran within a second. The issue occurred when trying to compute groups of three and four, that’s when the runtime became unacceptable. I will explain my optimized code in an example…



Being able to eliminate values by looking at old computation significantly reduces run time, because I am not rerunning through all possible groups each time. As I am finding a large set of groups, I can eliminate ones that didn’t already meet the support threshold and not compute them again. Figuring this out was a major milestone in my project and answered my question of making my code efficient. My third and final step was computing confidence scores of each group and each item in the given dictionary. This part was rather tedious since it was more figuring out the proper formula to compute each confidence. It was straight forward for finding groups of two, but for groups of three and four it became much more difficult. Below I have presented pseudo-code of my main algorithm for finding groups of items.



As mentioned before the dataset contains 38,765 rows and three columns. Each row contains a member number, date, and item description. I retrieved this data set from <https://www.kaggle.com/datasets/heeraldedhia/groceries-dataset>. The main challenge with this data was filtering out needed and unnecessary data. It took some time to learn how to get exactly what I needed. In this dataset I had no use for the date of purchase column. It added no value to my results. The metrics I used to evaluate my outputs were using float types and strings. For each string of items bought together there would be a whole number of times that item was bought. Along with each string of items there was a decimal number to represent the confidence scores of each one. I think metrics used are justified, it is given in a simple format that any can easily read. You can scan the results for a specific group of items and find how many times it occurred and the confidence score of it.

I will have about six different graphs to show for my outputs. I presented the groups of twos, threes, and fours in my output. For the sake of keeping the graph from being cluttered I graphed the top 10 results of each. Here is what each looked like…

Calendar

Description automatically generated

For the graphs above I would recommend viewing my Final.ipynb that includes these graphs plus more in better quality. As depicted above I have six different figures of results. The first two depict my top 10 results for pairs, next two are for triples, and the final two are for quads. Origally I had wanted to print out all my results, but that wouldn’t have worked. It would’ve been a very difficult graph to read. Instead, I decided to show the top 10 results, as it shows the general trend of my data. To view the entire output of each graph I have printed the outputs to a text file call final.txt.

During my time working on this project I learned two very valuable lessons. The first on being it is worth the effort to draw out your ideas and think of an algorithm before going straight to the code. I found myself stuck writing bad code, and only began to make real progress once I stepped back and made myself a plan. The second major thing I learned was how to optimize my code. I have never in my other classes had issues of my code running too slow. This project really made me must step back and think outside the box. It was difficult for me to write code that technically worked but was not efficient; and then must think of a new and better way to write it again. I think that lesson will be something that I will carry on throughout the rest of my schooling and into my career. Just because your solution works doesn’t mean it is the one to settle for. A key decision I would make to further the project is spend more time on the whiteboard planning rather than just throwing myself into it and expecting it to work perfect.

SOURCES:

<https://www.kaggle.com/datasets/heeraldedhia/groceries-dataset>

<https://matplotlib.org/stable/tutorials/introductory/pyplot.html>

<https://pandas.pydata.org/docs/>