Alec Broughal
December 12, 2023
CIS 635

Python Pipeline for Advertisement Recommendation Based on Collected Nike User Data

Introduction

For my final project, I chose to create a pipeline around recommending an advertisement based on user shopping data from an inputted CSV file. It is crucial for companies to get their advertisements in front of users who are the most likely to click on those advertisements to then purchase what is being recommended to them, and that is what the pipeline I wrote does. I have made numerous purchases purely based on advertisements that I have seen, and I wanted to be on the other end of that, where I am the one taking in data and recommending the appropriate advertisement per user. I want to work for Nike upon graduating, so I had personal motivation for choosing a project that was centered around them. I also wanted to work alone on this project so that I could work on every aspect of the pre-planning and actual coding to deepen my understanding of all of the moving parts and build up on my Python coding skills.

The pipeline that I wrote takes an input CSV file of user data for 500 users in the United States and then sorts through it to recommend an advertisement for each user that they are most likely to click on, and outputs a new CSV file of the recommended advertisement per user. The pipeline takes into account that for this data set, it takes place in the first quarter of the year, which is typically the coldest time of the year in the United States. It looks at which state the user is from to make an appropriate recommendation in case there's a tie between different clothing items that the user viewed and has a ranking system for tie breakers. The results from this project is a newly created CSV file that contains the first name, last name, email address, and the recommended advertisement per each user.

Related Work

Prior to this project, I had not coded anything on this scale or with as much freedom as I had on this project. I don't have any related work aside from Python homework assignments from previous classes that I've taken. I spent a while before the coding even began trying to think of the best way to go about writing this pipeline, and had to make several revisions to the plan of action before settling on the final plan that I submitted earlier this semester. Once the coding began I had to tweak it in some places so that the pipeline could operate, as writing it on paper is different than what actually gets coded.

Looking at other academic publications helped me to see how a similar functioning pipeline worked on a much larger and professional level, and also showed me how others go about planning and preparing for a project like this. Article 1 down in References assisted me with breaking down the issue I wanted to solve, and I used it as a guideline for creating my initial

plan of action as I liked the way it started off with the problem statement and how the authors were going to go about resolving the issue.

Section III-A from article 4 (down in References) implemented a tree model approach for advertisement recommendations, with the user being the root node. It then branched down into three main categories, each with subcategories and then subcategories underneath those, and kept going until every option was covered. I knew that I couldn't implement that same tree model for my pipeline as the main category was already selected (upper body clothing) and there were only seven options within this category, so the tree model would have been extremely small and not helpful to me at all for this specific project. Seeing how the tree model was implemented in this pipeline helped me in narrowing down my own project, even though I didn't go the same route that the authors did.

Article 2 down in references had good visualizations of real-world architecture for MAR-CF systems, such as Figure 4 on page 6 of that article. MAR-CF systems are different from what my pipeline is accomplishing, but were still valuable to me in helping me create and structure my pipeline. Table 2 did a good job displaying the results of the pipeline and I used it as a model for how I wanted my results to look, even though mine were on a CSV file of 501 (1 header and 500 users) and this just had 5 lines.

Methods

The user data that I found is not real, as Nike does not publicly share that information. Instead, I found a free CSV of fake data (article 3 in the References section) for 500 users that I then went into and modified to better fit this project. The data per user includes their first name, last name, city, state, Nike account email, year quarter, clothing size, as well as the 10 most recent upper body clothing items that they viewed on Nike's website. I removed unnecessary data columns from the CSV that didn't contribute to the overall project. At random, I added the more relevant data, which included the year quarter, clothing size, and the 10 shopping items each user clicked on while on Nike's website. For the sake of simplicity and also time, I chose to stick to only upper body clothing, which included seven different types of upper body clothing: polo shirt, short sleeve shirt, long sleeve shirt, button down shirt, crewneck sweater, hoodie, and zip up jacket. Based on which products each user viewed, the pipeline recommends one out of the seven clothing categories that it believes the user is most likely to click on when they see an advertisement for it on other websites.

Once I had the modified CSV of user data, the pipeline was able to begin. It starts off by asking the user that is utilizing this pipeline to choose a CSV file to be inputted. Once inputted, the first thing the pipeline does is copy the first row, which is the header row, from the CSV to a new Python file so that it can see exactly what the headers are to know the columns where the specific information lies. It then turns the headers into strings, and then they are turned into lines, and then finally they are added to individual lines into another Python file. Once the pipeline knows which column includes what data, it searches for the column for the year quarter to help rank the clothing items when making the recommendation. For this project, I chose to have the data in Q1 (January-March) which is typically the coldest quarter of the year. This will be used further down the pipeline.

A different Python file breaks up the 50 US states plus Washington DC into three weather categories; warm, mid, and cold, and the states are added into one of those three

categories. I categorized the states myself by looking at a map of the US and put each state in the category that I believed to be most fitting. For example, Alaska is added into the cold category, Tennessee is in the mid category, and Florida is in the warm category.

The pipeline then looks at the inputted CSV and finds which column is the state column, and depending on which state the user is from, it adds one of the three weather categories to the user and writes that to a new text file. It then searches through the CSV for the ten clothing items that they viewed most recently and adds those ten items behind the weather categories, with each value in the row being separated by a comma. It does this for all 500 users.

A different Python file reads this text file and counts the total of each item per user. If a user has viewed one clothing item more than all of the others, the pipeline writes that item as the recommendation to a new text file and then moves onto the next line for the next user. If there is a tie between two or more clothing items that were each viewed the same number of times, the pipeline writes to that same text file, but it first includes the weather category that the user is placed in, and then it adds all of the clothing items that were tied with each other, with these all being separated by commas. This gives us an almost ready recommendation file, as some users have a clear advertisement recommendation based on their viewed item habits, but others users have ties.

A different Python file reads this text file and if the first value is not one of the three weather categories, it writes that item to a new text file, which is the final recommendations text file. Since the user has viewed that item more than any of the other items, the pipeline recommends it as their advertisement rather than recommending a clothing item that they were less interested in. If the first category is one of the three weather categories, it searches for the corresponding text file for that weather category which has a ranked order of clothing item importance based on geographic location. Since it is Q1, the weather is much colder this time of the year in the United States, and so a user is more likely to click on a hoodie or jacket advertisement than a short sleeve t-shirt advertisement. But this is also based on where in the United States the user is located, as Georgia and Wisconsin have very different temperatures during this first quarter of the year. This is why the states are divided up, and the three categories have different ranked orders of importance for the clothing items. Once the pipeline finds the corresponding weather category text file, it checks to see if any of the tied clothing items are the first item in that text file, as that is the more important clothing item to recommend. If it is, then that clothing item is what gets recommended and gets added to the final recommendations text file. If none of the tied clothing items are the top ranked item, it moves on to the second ranked item, and does so until one of the clothing items becomes the first match, and that is what gets added to the final recommendations text file.

Now that we have this text file with the final advertisement recommendations for each user, there is one last Python file that takes this recommendation and adds it to a newly created recommendations CSV along with the first and last name of the user and their email address associated with their Nike account, so that the advertisement agencies can place the correct recommended ad in front of the user most likely to click on it and make that purchase.

I used Visual Studio Code to write the code for this pipeline in, and installed the Rainbow CSV, Python, Pylance, & Increment Selection extensions. These allowed the pipeline to read the CSV and use Python functions to run through the imported CSV and read the text files that the pipeline created and used.

Results and Discussion

I am very pleased with the results of this pipeline. It takes a CSV of 500 users each with 18 different columns of data and sorts through them all to find which column has what information in it. The columns could all be in a scrambled order each time the pipeline is run and it will still find the information it needs, regardless of the order of the columns. It is then able to move forwards with calculating which advertisement to place in front of the user so that they're most likely to click on it over the other advertisement options. The results are added to a newly created CSV file and placed within the same folder, ready to send out to advertising agencies to get the correct ad in front of the correct user.

I've included three images below, Image 1 being of the inputted CSV and Image 2 being the results CSV that is created with the recommended advertisements. Image 3 is of the pipeline's main.py function that calls everything together, so that the project is as simple as hitting Run, typing in the name of the CSV file that we results from, and then receiving a CSV file created with our results, as seen in the second image.

Going through some of the results, starting on Line 2 of Image 1, we see in columns H-Q that the 10 items this user viewed are 2 hoodies, 1 zip up jacket, 3 long sleeve shirts, 2 crewneck sweaters, 1 short sleeve shirt, and 1 button down shirt. As the long sleeve shirt had the highest view count, the pipeline did not need to look at the state this user is from and the year quarter, and so the pipeline recommended a long sleeve shirt advertisement, as we see in Line 2, column D of Image 2.

Checking the results of a different user, we can look at Line 13 in Image 1, and see that this user clicked on 2 polo shirts, 3 long sleeve shirts, 3 zip up jackets, 1 hoodie, and 1 crewneck sweater. Since this user had a tie between the long sleeve shirt and the zip up jacket, the pipeline first checks which time of year it is by viewing the Year Quarter, which is in column F in Image 1. Seeing that it's Q1 (January-March), it is the coldest time of the year in the United States and this information is used to help rank the clothing items. The pipeline then looks at the state the user is from, which is Column D in Image 1, and this user happens to be from New Jersey, which is pre-categorized into the coldest weather category. Using this information, the algorithm determines that a zip up jacket would be more useful over a long sleeve shirt this time of year in that state, and so the zip up jacket is what gets recommended in Line 13, column D of Image 3.

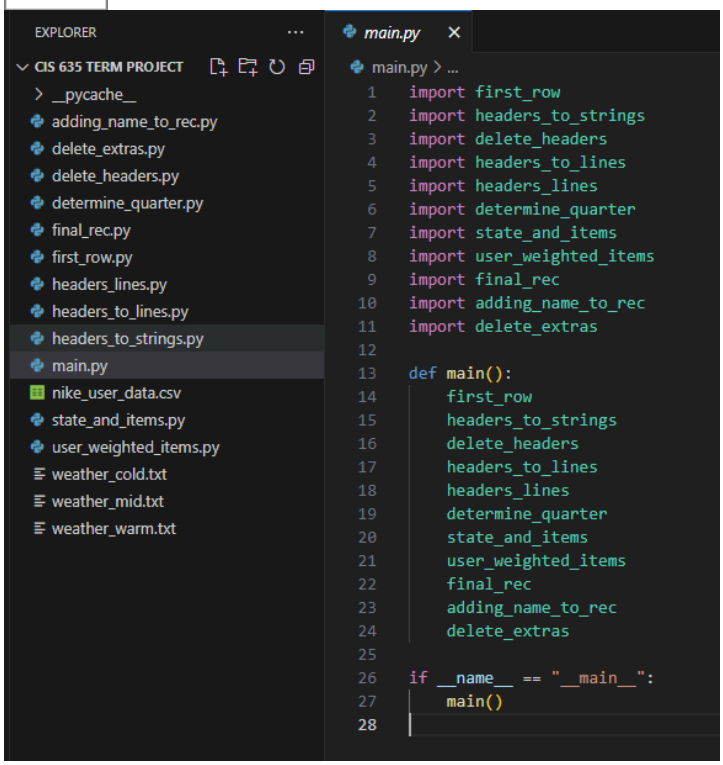
Image 1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	First Name	Last Name	City	State	Account Er	Year Quart	Size	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	
2	Abel	Maclead	Middle Isla	NY	amaclead(	Q1	S	Hoodie	Zip up Jack	Hoodie	LS Shirt	Crewneck	SS Shirt	Crewneck	LS Shirt	LS Shirt	Button Down Shirt	
3	Adelina	Nabours	Cleveland	OH	adelina_na	Q1	S	LS Shirt	SS Shirt	Crewneck	LS Shirt	Hoodie	Button Do	LS Shirt	Crewneck	Hoodie	LS Shirt	
4	Adell	Lipkin	Whippany	NJ	adell.lipkin	Q1	L	LS Shirt	LS Shirt	SS Shirt	SS Shirt	SS Shirt	SS Shirt	SS Shirt	LS Shirt	Hoodie	SS Shirt	
5	Ahmed	Angalich	Harrisburg	PA	ahmed.an	Q1	M	LS Shirt	Crewneck	Hoodie	LS Shirt	Button Do	LS Shirt	LS Shirt	Hoodie	LS Shirt	LS Shirt	
6	Aja	Gehrett	Nutley	NJ	aja_gehret	Q1	XL	SS Shirt	Zip up Jack	LS Shirt	Hoodie	SS Shirt	LS Shirt	Polo Shirt	Crewneck	SS Shirt	Button Down Shirt	
7	Alaine	Bergesen	Yonkers	NY	alaine_ber	Q1	XL	SS Shirt	Crewneck	LS Shirt	LS Shirt	Zip up Jack	LS Shirt	Crewneck	SS Shirt	LS Shirt	LS Shirt	
8	Albina	Glick	Dunellen	NJ	albina@gli	Q1	M	Hoodie	LS Shirt	Button Do	Hoodie	LS Shirt	Zip up Jack	Hoodie	LS Shirt	Zip up Jack	SS Shirt	
9	Alease	Buemi	Boulder	CO	alease@bu	Q1	XS	SS Shirt	Crewneck	Hoodie	SS Shirt	Hoodie	LS Shirt	Hoodie	LS Shirt	LS Shirt	LS Shirt	
10	Alecia	Bubash	Wichita Fa	TX	alecia@ac	Q1	S	Polo Shirt	Zip up Jack	Zip up Jack	LS Shirt	LS Shirt	Button Do	LS Shirt	Hoodie	Button Do	Button Down Shirt	
11	Alesia	Hixenbaug	Washingto	DC	alesia_hixe	Q1	S	Hoodie	Crewneck	LS Shirt	SS Shirt	Button Do	SS Shirt	SS Shirt	Crewneck	SS Shirt	Zip up Jacket	
12	Alex	Loader	Tacoma	WA	alex@load	Q1	L	SS Shirt	LS Shirt	LS Shirt	Hoodie	SS Shirt	LS Shirt	Button Do	Hoodie	LS Shirt	Polo Shirt	
13	Alisha	Slusarski	Middlesex	NJ	alisha@slu	Q1	S	Polo Shirt	LS Shirt	Zip up Jack	Hoodie	LS Shirt	Zip up Jack	Crewneck	Zip up Jack	Polo Shirt	LS Shirt	
14	Alishia	Sergi	New York	NY	asergi@gm	Q1	S	SS Shirt	LS Shirt	LS Shirt	Crewneck	Hoodie	LS Shirt	Hoodie	Button Do	Button Do	Hoodie	
15	Aliza	Baltimore	San Jose	CA	aliza@aol.	Q1	S	Hoodie	Crewneck	LS Shirt	LS Shirt	Crewneck	Zip up Jack	Zip up Jack	Zip up Jack	Zip up Jack	SS Shirt	

Image 2

	A	B	C	D	E
1	First Name	Last Name	Account E	Ad Recommendation	
2	Abel	Maclead	amaclead(	LS Shirt	
3	Adelina	Nabours	adelina_na	LS Shirt	
4	Adell	Lipkin	adell.lipkin	SS Shirt	
5	Ahmed	Angalich	ahmed.ang	LS Shirt	
6	Aja	Gehrett	aja_gehret	SS Shirt	
7	Alaine	Bergesen	alaine_ber	LS Shirt	
8	Albina	Glick	albina@gli	Hoodie	
9	Alease	Buemi	alease@bu	LS Shirt	
10	Alecia	Bubash	alecia@ac	LS Shirt	
11	Alesia	Hixenbaug	alesia_hixe	SS Shirt	
12	Alex	Loader	alex@load	LS Shirt	
13	Alisha	Slusarski	alisha@slu	Zip up Jacket	
14	Alishia	Sergi	asergi@gm	Hoodie	
15	Aliza	Baltimore	aliza@aol.	Zip up Jacket	

Image 3



Conclusion

This project took me around 50-60 hours from start to finish, from coming up with the idea for the pipeline to writing down an initial plan of action to executing that plan. For the most part, I was able to follow my plan of action almost in its entirety, but I did run into some issues that made me change the project in some ways. Initially I had hoped for a more specific sorting method for choosing the recommended clothing item, but as I don't have much coding experience in Python, I was not able to get it coded as precisely as I had initially hoped.

The initial idea was to really look at the order of the viewed clothing items per user on top of the total count of each item. If any one item had a higher count than the rest, that was to be the recommended item for that user, which I was able to do successfully. If there was a two-way tie between two items, the code checked the weather category that the user was in and chose the item that was ranked higher in that category, which I was able to do successfully.

If there was a three-way tie I wanted to code it to check if one of the three items was viewed in complete consecutive order. So if the short sleeve shirt, crewneck sweater, and zip up jacket all had a count of 3 and the hoodie had a count of 1, the pipeline would then check if one of the three were viewed all three times consecutively. If that was the case, the pipeline would recommend that item for the advertisement. If two or all three were viewed completely

consecutive, then whichever of the three was viewed closest to the end of the list would be chosen. Otherwise if none were completely consecutive then it would look at the weather category for that user and choose the top ranked item. The pipeline would do the same calculations for a four-way item tie. I was not able to code the pipeline to follow all of these conditions, and it reached a certain point where I had to change my initial plan of action for something that was more within the lines of my coding abilities.

If there was a five-way tie between clothing items, I initially wanted the pipeline to check if one of the items was both the first and the last item viewed, and if so, then that would be the recommended item. Otherwise if that wasn't the case, it would check if any of the five were viewed consecutively and if only one was, that would be the recommended item, otherwise if two or more were viewed consecutively, the consecutively viewed item closest to the end of the list would be chosen. And if none of the five items were viewed consecutively, then the pipeline would check the weather category for that user and make the recommendation based on that ranking. I also was not able to code this tie breaker the way I had initially intended to in my plan of action.

In the end, I was able to code the single highest counted item and also the two-way tiebreaker items, but the three-way, four-way, and five-way ties were not able to be coded as I had initially planned to, and instead they are coded the same way as the two-way tie. I am glad that I was still able to implement the use of the weather categories, as there was a point during the coding of this pipeline where I wasn't able to get it reading the ranked clothing items at all and I thought that I was going to have to eliminate that portion entirely, but I was able to make it work in the end.

I plan to continually improve on this project in my personal time moving forwards, as I know that it has a lot more potential for what it can do. One of my plans that I did not have time to implement for this project is the weather categories for all four quarters of the year, not just the first quarter. The summer ranked items would be very different from the winter ranked items, and it would be more accurate and for the pipeline to be able to work with all four quarters, rather than just Q1 since I knew that this was the quarter we were dealing with.

Another extension to this project would be to add more clothing items as opposed to the seven that I implemented, and to add more than ten viewed items per user. It would also be more realistic for each user to have a different number of viewed items. I think that capping it off on 20 items would be more reasonable, but not every single user clicks on that many items, so one user could have only 9 items while the next one has 20 and the user after that has 11. It would change up how it's coded since right now the function is looking at all 10 items and would probably crash if there was an empty column for a user who viewed under 10 items.

A third possible project extension would be to have more than just three weather categories, as more than half of the country was placed within the Cold category. And if all four quarters are taken into account, the states in each category could vary, as the seasons aren't the same per each state year round and I could be a lot more precise with the recommended advertisement per user.

Data and Software Availability

Article 5 in References is the link for accessing this final project. It contains all of the files necessary for this pipeline to operate. I would create an empty folder on your local computer to

place all of the downloaded files within. If you do choose to run this pipeline, I recommend running it in Visual Studio Code as that is where I coded the entire project and know for certainty that it will run successfully. Please note that you will need to have the Python, Pylance, Rainbow CSV, and Increment Selection extensions installed in Visual Studio Code for this pipeline to run properly. The four extensions are all free and can be downloaded and installed from directly within Visual Studio Code. There are 12 Python files and three text files that make up what is needed for this pipeline to operate. The CSV of the user data is also included in the GitHub Repository, so a total of 16 total files. The Pipeline creates numerous temporary text files and Python files throughout the project that are not necessary at the beginning nor end of the pipeline, but I have included those files in the Repository as well just in case the reader is unable to run the project in Visual Studio Code and wants to see the content of those files. The last Python file that the pipeline runs goes through and deletes these temporary files so that the only addition to the folder is the newly created Recommendations CSV file.

References

1 - “Adaptive Advertisement Selection and Recommendation System Based on User’s Browsing Style.” *International Journal of Advanced Trends in Computer Science and Engineering*, 2013, citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c76c3fac72314481a0c41cb8f562ecdd0949567d

2 - Ahn, Hyunchul. “Mobile Advertisement Recommender System Using Collaborative Filtering: MAR-CF.” *KAIST*, koasas.kaist.ac.kr/bitstream/10203/5362/1/2006-063.pdf.

3 - Dunning, Brian. *Free Sample Data for Database Load Testing*. https://www.briandunning.com/sample-data/

4 - Kang, Seongju, et al. “Tree-Based Real-Time Advertisement Recommendation System in Online Broadcasting.” *IEEE Xplore*, 19 Oct. 2020, ieeexplore.ieee.org/abstract/document/9229171.

5 - Broughal, Alec. GVSU CIS635 Term Project Team Broughal. https://github.com/GVSU-CIS635/gvsu-cis635-term-project-team-broughal