UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE POLITICHE, ECONOMICHE E SOCIALI

DIPARTIMENTO DI ECONOMIA, MANAGEMENT E METODI QUANTITATIVI

Master degree in
Data Science for Economics

MARKET-BASKET ANALYSIS

Student:
Daria Shcherbakova
17487A

ACADEMIC YEAR 2023-2024

Contents

Chapter 1

Introduction

The main goal of the firm is to maximize its profit. One of the common ways to achieve this aim is to force customers to spend more money, meaning that the business should make consumers buy more products. For this purpose, the business owner could suggest which (extra-)items the customer should buy, suggesting goods that could be relevant for the consumer, based on the usual basket of the consumer. Thus, the market-basket analysis allows retailers to identify relationships between items that people buy. It works by looking for combinations of items that occur together frequently in transactions. The idea is to answer the question: how likely will a customer buy the product A if the customer already has the product B in their shopping cart?

However, frequent items analysis could be performed not only on customers' data but also on different types of data corresponding to market-basket data. In other words, we can apply market-basket analysis to any data consisting of two kinds of entities (items and baskets), representing many-to-many relationships. In this project, the market-basket analysis is implemented on jobs and skills from LinkedIn. The main goal is to find frequent job skills that occur together. Therefore, it may help a candidate to understand which skills he or she should gain, based on already gotten ones, to apply for a particular position, because they are frequently and commonly asked by recruiters. For this purpose, the data has been sampled for jobs related to *data science*, *data analysis*, and *data engineer* fields. To find frequent items (job skills) the A-Priori algorithm has been performed with the PySpark framework.

Theoretical background

Let us introduce the main definitions and concepts to understand the project better and perform algorithms. Firstly, it is important to mention that the data should be presented in a particular format to perform market basket analysis. It should

represent the many-to-many relationship between 2 kinds of objects: items (skills in our case) and baskets (job advertisements). The data by itself is represented as a sequence of baskets. Each basket consists of a set of items (constructing itemset). We should control that the number of items in a basket should be small, greatly less than the total number of items.

In this analysis we focus on the frequent set items, meaning that occurs in many baskets. To identify the *frequency* we use the terms of support and support threshold. So, ***support*** for the set of items can be identified as the number of baskets for which this set of items is a subset. In other words, it is the frequency of the set of items among all baskets. In this case, ***support threshold*** is the particular drop-step after which a set of items could or could not be considered frequent (it should be reasonably high to escape not-really frequent itemsets (around 1% of the baskets), which does not have the meaningful influence on the outcome). We should control the number of frequent itemsets and adjust support value because it directly leads to algorithm efficiency.

Another important concept is the ***association rule*** (if-then) referring to the fact that if all items of the itemset appear in some basket, then the particular item is likely to appear as well. This likelihood can be identified as ***confidence***, the ratio of support for the itemset and the item to the support for the itemset. Equally, it is a fraction of baskets (job advertisements) with all items of the itemset (a set of job skills) that also contain the item (a particular job skill). Confidence measures the strength of association between two itemsets (it should be around 50%). However, in this project the strength is ignored as well as the direction of association (which is usually measured with *confidence, lift*).

The idea of the base algorithm to perform MBA is like this: it receives the list of sets of items to find those, whose frequency is above the support threshold. And then we work with all possible combinations of frequent items to find the most frequent set of items based on the support threshold. Each time we generate a pair, we add 1 to its count. In the end, we execute all pairs to see which have counts greater or equal to the support threshold to consider them as frequent. However, if there are too many pairs, the algorithm may fail, and each algorithm has a limit on how many items it can deal with. Since the algorithms require maintaining many different counts by each pass through data, it is more space-efficient to represent items by consecutive integers, for that hashtable may be used. Also, for more efficiency, we can use Map-Reduce to divide the process among many processors.

In this project, I implement the A-priory algorithm, which is designed to reduce the number of pairs that must be counted at the expense of performing 2 passes over the data (to skip the step of storing not frequent items) based on ***monotonicity of items***. This rule says that if a set A of items is frequent, then every subset of A is frequent, helping to compact the information about frequent items.

At the first pass we simply translate item names into integers (enumerating each observation with key-value pair with map-encoding), count the frequency of each item and associate it with the item stored in the memory. Basically, as soon as we translate skills into integer representation (similar to a dictionary), we use it to index in the array of counts and add +1 to the integer found. After this step, we ask to store only those items higher than the support threshold value. The second pass serves to create new integers for frequent items only, which we have stored after the first pass, creating a frequent item table. Here we count all the pairs that consist of 2 frequent items (remember that a pair cannot be frequent unless both its members are frequent due to the monotonicity rule). We are considering all possible combinations of frequent items after the first pass to check them for being frequent itemsets according to the support threshold value.

Chapter 2

Data, data preprocessing, sampling

In this project, I used a dataset from Kaggle, focusing primarily on the *job-skills* file and supplementing it with the *linkedin-job-postings* data for sampling purposes. While the initial job-skills dataset was extensive, necessitating sampling or alternative processing techniques, I opted to leverage job posting details to create a subset of relevant jobs.

Recognizing the importance of representative sampling, I avoided selecting data from a single segment of the large file, as this could lead to skewed results due to potential data inconsistencies within the file itself. To ensure valid and insightful analysis, I established specific sampling criteria. This included targeting job levels (*Mid-Senior*), searching within the *United States*, and focusing on positions related to *data science, data engineering, data analysis, and business analysis*. While computational efficiency played a role, my personal interest in these fields also influenced the sampling approach.

This targeted sampling resulted in a subset of approximately 7000 data points (less than 1% of the original data). However, this subset offers a significant advantage: it allows for analysis of complete job descriptions within a specific field, country, and experience level, leading to more insightful, representative, and valid results. Following this selection based on LinkedIn job postings, I then filtered the main dataset.

Given the *many-to-many* relationship within the data (job postings as baskets and skills as items), we treated the data accordingly. Furthermore, recognizing that the data originated from real-world sources and potentially contained text inconsistencies, I implemented several data preprocessing steps. This included converting text to lowercase, tokenization, stopword removal, and lemmatization to enhance data representation. The initial data contained irregularities, and this preprocessing stage helped to eliminate duplicate counting of words with similar

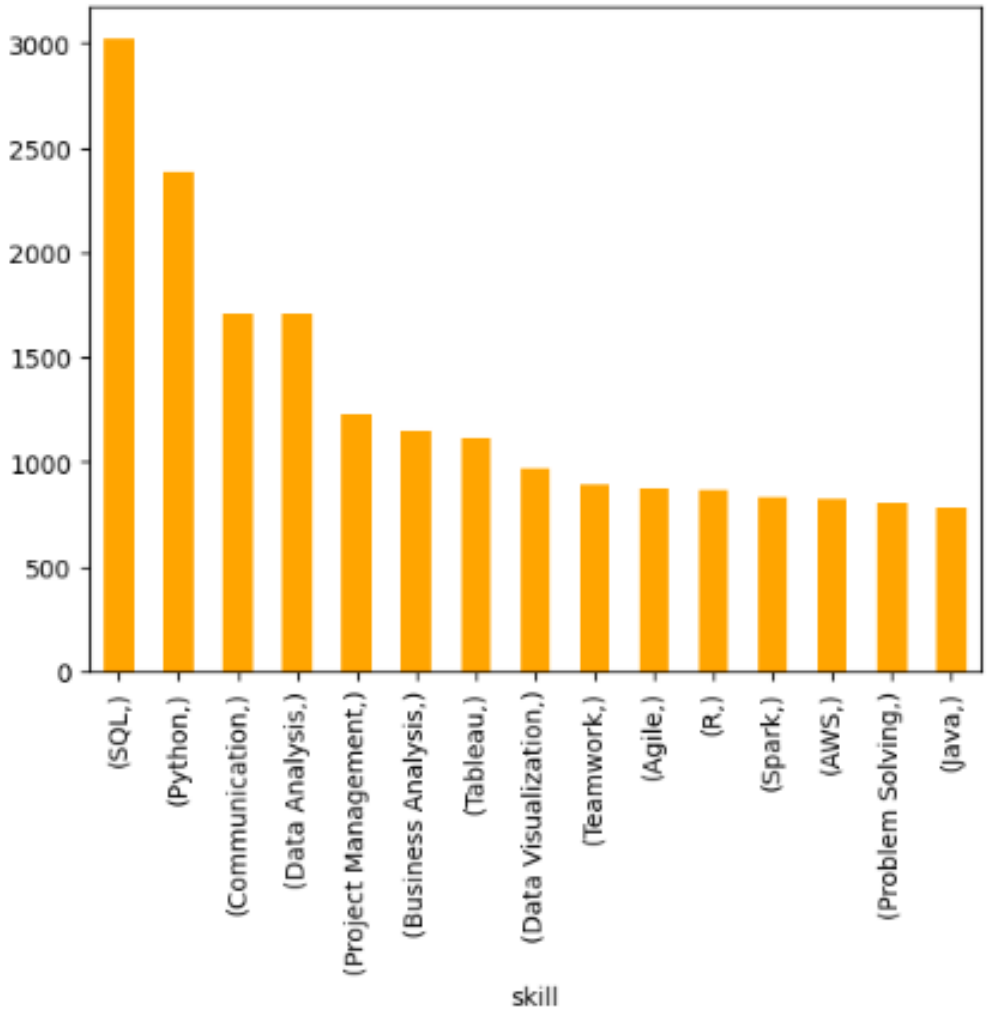

Figure 1: EDA: frequency of skills

meanings but different spellings. Perhaps, it was not necessary to dive in removing stopwords, salary, or experience information due to potential infrequency, these steps were ultimately deemed unnecessary.

The final working dataset comprises 6795 rows with two columns. However, for analysis purposes, we only require the skills column, not the links. The Apriori algorithm was then applied to both the preprocessed and the extra-preprocessed data for further analysis.

From the basic explanatory data analysis, I can say that I may expect to see frequent skills such as SQL, Python, Communication, Data Analysis, Project Management.

Chapter 3

Frequent Items algorithms implementation

3.1 Data preparation for MBA

Market Basket Analysis is based on the idea that we have a list of set of items. Among these we should find those, whose frequency is above the threshold (support), in other words, finding the most frequent set of items is the process of finding possible combinations.

For the project I used the Functional programming paradigm, the most importantly code chunks are written in functions mode for efficiency and ease of use. Before implementing the algorithm, we transform our data (strings with sequences of skills) into baskets. For this purpose we select the column dedicated to skills and transform it into an RDD (resilient distributed dataset) data structure for PySpark, splitting each skill by comma-separator. This function returns the list of lists with elements corresponding to a single skill. So, for the preprocessed and extra-preprocessed datasets, I have: the maximum of items in the basket is 224, the average number of items in the basket is about 27, and the total number of baskets corresponds to the dataset size.

After this step, we can calculate and assign the support threshold value. As I have mentioned earlier, it could be calculated as 1% of baskets number. However, it could be rearranged based on the algorithm efficiency and data (that is why I made a flexible function with the opportunity to insert the desired percentage). In my case, the support threshold value is equal to 68, meaning that the skill may be considered frequent only if it appears in more than 68 baskets.

The next important step is to create hash-table, which we use to translate skills into an integer representation (via hash function) for its efficiency. Hash-tables by themselves are considered one of the most useful basic data structures, providing

O(1) complexity. Here we input the list of lists (our baskets) with skills and extract each skill, transform into a single flat list containing all the unique skills from all baskets. After this, we remove all duplicates and generate index mapping (creating an index to for each unique skill stored in tuples data structure). Next, these skill-index pairs in tuples representation are translated into a dictionary (hash map), where the key is the unique skill, and the values are their corresponding indexes in the original flattened list. The function returns hash map that maps each unique skill to its index in the items list. We have to apply this function for all baskets iterating each skill in the basket to create the dictionary for all possible items in all considered baskets. Thus, the total number of skills is 41020 for the preprocessed dataset and 35318 for extra-preprocessed one.

3.2 APriori Algorithm

Because the sampled data has a final size of around 7000 rows, the A-priori algorithm is suitable for this dataset. Here I'm focusing on the base Apriori algorithm, as optimizations are typically targeted towards larger datasets like PCY and Multistage. Also, I have decided to follow the tyranny for counting pairs, expecting to find more frequent pairs than frequent 3,4,..-plets. The Apriori algorithm is designed to efficiently identify frequent itemsets by focusing on pairs first. This avoids unnecessary computations for larger groups (3-plets or more). I created a function that implements the two-phase Apriori algorithm. This function serves as input basketiresed data, support threshold, indexes from hash-table, and the size of set (in my case is 2). This entire function can be implemented to both datasets without changes.

In the first phase, we need to count items and select only frequent ones according to the support threshold. The code realization is this: for each skill (item) in the basket we create a tuple containing the skill and its count (initially it is equal to 1) and it iterates through each basket. After this, we perform ReduceByKey by grouping skills based on the key (which is the skill by itself) and summing corresponding counts. Based on the previous step we can filter and store only those skills that are frequent based on the threshold, in other words, it keeps only those items whose total count is greater than the support threshold. I have decided to return some information after this pass, and I ask to return the top-5 frequent skills ordering them in descending way and getting the name of skills from the previously created hash-table.

In the second pass, we create and count frequent itemsets (again based on the support threshold value). The code is realized as follows: we take distinct frequent items and create the candidate pairs through each basket. Basically, we generate all possible pair combinations from the sorted skills within the basket and check if

all subsets (itemsets of size k-1) in the itemset exist in the set of candidate skills (from the first pass). After this, we apply ReduceByKey to group pairs, count their frequency, and filter those pairs, that are higher than the support threshold value. Once again I ask the function to return the top-15 frequent skill pairs.

Chapter 4

Results

After the algorithm converges (the runtime depends on the dataset size.), the output is top-5 frequent skills (our candidates) after the first algorithm's pass and top-15 frequent skill pairs (after the second pass).

In the case of data without extra-preprocessing, the number of skills that are candidates to be frequent is 352. Our top-5 frequent skills are SQL, Python, Communication, Data Analysis, and Project Management. After the second pass, we have 1657 frequent skills pairs. While the results are a bit different for extra-preprocessed data (also from a timing point of view). So the total number of frequent skills is 360 (higher than in the previous case) and the top-5 skills are SQL, Data Analysis, Python, Communication, and Project management. Both results are expected due to primary EDA (fig.1.). The total number of frequent skill pairs is 2471. We can see that the order (frequency) of skills changes based on the extent of data preprocessing. Moreover, it may be the case that tokenization and lemmatization processes help to increase the frequency of particular skills due to writing them in the same way.

The frequent skill pairs are presented in Table 1, and there are some differences in terms of frequency, and some skill pairs are not presented in both columns (for instance, Python - Java, and Python - Tableau). Thus, the extent of data preprocessing leads not only to the order of skill pairs but also to the presence or absence of particular pairs.

Furthermore, it is important to mention that it takes around 4 minutes to perform the algorithm on preprocessed and basketerised data, while on extra-preprocessed and basketerised data it takes around 9 seconds. Therefore, the correct data cleaning and preprocessing are crucial before applying any analysis to get accurate results.

Freq.	**Preprocessed data**	**Extra-preprocessed data**
1.	Python - SQL	Python - Sql
2.	SQL - Data Analysis	Data Analysis - Sql
3.	SQL - Tableau	Sql - Data Visualization
4.	Python - R	Machine Learning - Python
5.	Python - Spark	Data Analysis - Communication
6.	Communication - SQL	Sql - Tableau
7.	Python - AWS	Python - R
8.	R - SQL	Python - Data Visualization
9.	Python - Java	Data Analysis - Data Visualization
10.	Python - Tableau	Data Analysis - Project Management
11.	Data Visualization - SQL	Data Analysis - Python
12.	Python - Machine Learning	Python - Spark
13.	Communication - Data Analysis	Sql - Communication
14.	Python - Hadoop	Machine Learning - Sql
15.	Python - Scala	Project Management - Communication

Table 1: Frequent pairs for both datasets

Chapter 5

Conclusion

In this project, I have implemented Market Basket Analysis from scratch on LinkedIn job descriptions using the PySpark framework. Without deep data preprocessing, the entire analysis takes about 10 minutes, with the Apriori algorithm itself consuming 4 minutes. Data cleaning (lowercasing, tokenization, lemmatization) can significantly impact performance and results, not only in terms of running time, but also in terms of the final output - skill pairs.

Generally speaking, the observed results are expected by me, because it is common sense to have such skills as programming languages and their combinations with BI software. However, I could say that I did not expect to see any soft skills among top-5 skills or top-15 skill pairs.

One of the possible improvements could be considered the confidence and lift metrics to control the strength and direction (relevance) within associations. For instance, on the base dataset, we have the frequent skill pair as SQL - Data Analysis, while on extra-preproccessed we have Data Analysis - SQL. Moreover, it may be useful to ignore the tyranny of counting pairs, however, there is a risk that the algorithm would take much time to converge.

This work shows the importance of data cleaning and preprocessing before any analysis should be implemented since it affects the possible insights that could be extracted from the data.

Appendix A

Appendix

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work, and including any code produced using generative AI systems. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

Bibliography

[1] Anand Rajaraman and Jeffrey D Ullman. *Mining of massive datasets*. Autoedicion, 2011.

[2] Aditya Y Bhargava. *Grokking algorithms*. Simon and Schuster, 2024.

[3] IUYasik. Market basket analysis: Apriori algorithm using zhang's metric. URL:https://medium.com/@iuyasik/market-basket-analysis-apriori-algorithm-using-zhangs-metric-708406fc5dfc, 2023.

[4] Rdd programming guide. URL:https://spark.apache.org/docs/latest/rdd-programming-guide.html.

[1] [2] [3] [4]