Institute for Visualization and Interactive Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis

Visualizing the Interactions and Relationships from Sales Data and Data-Driven Automatic Product Bundling to Increase Cross-Selling

Vinoth Kannan

Course of Study:	M.Sc. Information Technology
Examiner:	Prof. Dr. Daniel Weiskopf
Supervisor:	Daniel Klötzl, M.Sc., Akhileshwar Sharma, M.Sc.
Commenced:	January 16, 2023
Completed:	July 16, 2023

Acknowledgement

I would like to convey my sincere gratitude to Professor Dr. Daniel Weiskopf for accepting the master thesis topic from the company and granting me the chance to work at the Visualization Research Center (VISUS) of the Institute of Visualization and Interactive Systems (VIS) at the University of Stuttgart. I am deeply indebted to my supervisors, Daniel Klötzl, M.Sc., and Akhileshwar Sharma, M.Sc., for their invaluable guidance and unwavering support throughout this thesis. Their profound expertise, thought-provoking discussions, and continuous encouragement have played a pivotal role in shaping this work and keeping me motivated on this challenging journey.

Furthermore, I would like to thank the Würth organization for providing me with an industry-oriented topic and for their support throughout the completion of my thesis. Their collaboration and assistance have been crucial in gaining practical insights and enhancing the relevance of this research.

I am grateful to all the individuals who contributed to this thesis project in various ways, including providing valuable input, participating in interviews, and offering their expertise.

Lastly, I would like to express my deepest appreciation to my family and friends for their unwavering support, understanding, and encouragement throughout this journey.

Abstract

E-commerce has emerged as a powerful platform for revenue generation in business. Various sales strategies have been adopted to maximize revenue while meeting customer expectations. Recent advancements in machine learning and deep learning have played a crucial role in enhancing e-commerce sales. This research explores the potential of employing machine learning techniques to provide personalized product bundle recommendations based on customer preferences, aiming to improve cross-selling and gross merchandise volume. However, machine learning models often lack explainability, which leads them to be termed black box models. Currently, there is a lack of explainable recommendation models that offer insights into the inner workings of deep learning models. To explore the product relationships and to understand and support the decision-making processes for different recommendation models, a visual analytics system is built. This thesis explains and interprets deep learning-based recommendations by visualizing the relationships and interactions within customer-product data. The visualization tool is built upon pilot interviews conducted from a business perspective to provide insights to stakeholders and enhance the interpretability of recommender systems. We explore the use of machine learning models for automatic product bundling, benchmark the results, and use visualization with the help of dimensionality reduction for high dimensional data, hierarchical edge bundling, scatter plots, collapsible trees, and heatmaps. The implementation involves employing JavaScript libraries, Python, HTML, and CSS. The visualization toolbox is built upon the requirements gained through a pilot interview. It was evaluated via an expert study that analyzed the visualization toolbox with parameters like transferability and evocativeness, reusability, abundance, and understandability.

Keywords: product bundling, recommendation systems, frequent itemset mining, collaborative filtering, alternate least squares, neural collaborative filtering, visualization toolbox.

Kurzfassung

Der E-Commerce hat sich als bedeutende Plattform zur Umsatzgenerierung in Unternehmen entwickelt. Um diesen zu maximieren und gleichzeitig die Kundenerwartungen zu erfüllen, wurden verschiedene Verkaufsstrategien entwickelt. Die jüngsten Fortschritte im Bereich des maschinellen Lernens und des Deep Learning haben eine entscheidende Rolle bei der Steigerung des E-Commerce-Umsatzes gespielt. Diese Arbeit untersucht das Potenzial des Einsatzes von maschinellen Lernverfahren, um personalisierte Produktbündel-Empfehlungen auf der Grundlage von Kundenpräferenzen bereitzustellen und gleichzeitig sowohl den Querverkauf und das Bruttowarenvolumen zu erhöhen. Allerdings mangelt es maschinellen Lernmodellen oft an Erklärbarkeit, was dazu führt, dass sie als Black-Box-Modelle bezeichnet werden. Derzeit gibt es einen Mangel an verständlichen Empfehlungsmodellen, die Einblicke in das Innere von Deep-Learning-Modellen bieten. Um die allgemeinen Produktbeziehungen untersuchen zu können und gleichzeitig die Entscheidungsprozesse für verschiedene Empfehlungsmodelle zu verstehen, wurde ein Visual Analytics-System entwickelt. Diese Arbeit erklärt und interpretiert Deep-Learning-basierte Produktbündelempfehlungen durch die Visualisierung der Beziehungen und Interaktionen innerhalb von Kunden-Produkt-Daten. Das Visualisierungstool basiert auf Interviews, die aus der Unternehmensperspektive durchgeführt wurden, um Einblicke für spezielle Interessengruppen zu liefern und die Interpretierbarkeit von Empfehlungssystemen zu verbessern. Wir untersuchen den Einsatz von maschinellen Lernmodellen für die automatische Produktbündelung, führen ein Benchmarking der Ergebnisse durch und nutzen Visualisierungstechniken wie Dimensionalitätsreduktion für hochdimensionale Daten, hierarchische Kantenbündelung, Streudiagramme, zusammenklappbare Bäumen und Heatmaps. Die Implementierung erfolgt unter Verwendung von JavaScript-Bibliotheken, Python, HTML und CSS. Die Visualisierungs-Toolbox basiert auf den Anforderungen, die durch ein Pilotinterview gewonnen wurden. Sie wurde mithilfe einer Expertenstudie evaluiert, die die Visualisierungs-Toolbox anhand von Parametern wie Übertragbarkeit und Anschaulichkeit, Wiederverwendbarkeit, Fülle und Verständlichkeit analysierte.

Schlüsselwörter: Produktbündelung, Empfehlungssysteme, Frequent Itemset Mining, Kollaboratives Filtern, Alternating Least Squares, Neuronales Kollaboratives Filtern, Visualisierungs-Toolbox.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem Statement

Companies seek innovative ways to improve customer satisfaction and boost sales revenue in today's competitive business environment. There are many sales techniques adopted to achieve it. For instance, customer churn prediction and customer segmentation will lead to customer-personalized sales. Customers who have purchased a certain product category could be recommended different products that they might pay attention to, which would increase revenue. Product bundling is one such method, in which the business bundles several products into a single package and sells it at a discounted price. This increases the gross merchandise volume and is used to encourage customers to buy more products. This strategy is gaining popularity, especially in the e-commerce sector, where customers are frequently provided with a vast selection of available products to choose from. Furthermore, for both customers and sellers, purchasing a bundle of products may be less expensive than purchasing each product separately. Benefits include an increase in average order value, a decrease in marketing and distribution costs, and a reduction in inventory waste.

To create successful product bundles, we must have a deep understanding of customers' preferences, product attributes, and pricing strategies. With the available big data, it will be harder to analyze and find patterns without machine and deep learning methods since it will take a lot of time and resources. Many businesses are using data-driven strategies that use sophisticated analytics and machine learning algorithms to study customer behavior and forecast their preferences in order to address this problem. So, exploring the possibility of machine learning and deep learning methods will not only be helpful in generating bundles but will also provide data-driven reasoning for the suggested bundles.

The basis for the product bundling algorithm is based on various recommendation methods and frequent itemset mining. We will compare outcomes from the methods, retrieve insights from them, and proceed with the best performing algorithm. The most commonly used recommendation algorithm in the industry is collaborative filtering [Sch07]. Even though it has disadvantages, its results are widely accepted. We will see variants of the collaborative filtering approaches like the Alternate Least Squares for personalized ranking [TT12] and the Neural Collaborative filtering (NCF) method [Sch07]. After applying these methods, we will benchmark the results to compare the two models trained with their respective hyperparameters and proceed with the best resulting algorithm. On top of it, using the results from the best algorithm, form a bundle with the customer's preferred products, making it a customer-specific bundle recommendation. After this, also derive the associative rules with the help of frequent itemset mining [SB17], for the purpose of finding frequent items that are purchased together and checking for bundling possibilities.

By utilizing visualization methods, we can enhance the decision-making process and enable interpretability of machine learning models. This makes significant importance from a business point of view since machine learning models are often considered "black box"models. By gaining

interpretability, businesses can ensure that customers are recommended products that are relevant, avoiding any mismatches or irrelevant suggestions. So, by applying the machine learning method, we can help in achieving interpretability with the help of visualization in order to provide the reason for the results that are predicted by the machine learning model. For this, we use a visualization study to conduct a pilot interview, build the interpretability tool based on the inputs, and finally evaluate the tools' effectiveness and purpose.

1.2 Motivation and Goal

This thesis will profoundly explore the different strategies for the product bundle using the recommendation algorithm and also support the reasons behind the product bundle results. For the purpose of supporting the reason for the results, we use the visualization toolbox that will be used by the business in order to understand the results. By using a visual analytics tool to reduce the high-dimensional data and provide various visualizations and knowledge graphs of the results, we will get more insights out of it. Therefore, the main research objectives are:

- How to automatically generate product bundles that are personalized to customers to increase cross-selling.
- How to visualize existing product relationships and interactions using the visualization toolbox.
- How to build a visualization tool that helps with the interpretability of the product bundling algorithm.
- Provide business KPIs visually and different hyperparameter settings and accuracy views of the chosen model.

As a result, the goal of the project is to provide a visualization tool to explain the decision-making process both for the domain experts and the management to increase cross-selling and satisfy customer needs through product bundling. Customers will be satisfied with product bundles if the products in the bundle are relevant to them and if they allow them to purchase the products of their choice in one transaction rather than searching for the relevant products separately. The business metric for deciding the impact of this project is the cross-selling turnaround due to the product bundling.

1.3 About Adolf Würth GmbH & Co.KG (AWKG)

1.3.1 General information on AWKG

AWKG was founded in 1945 as a screw wholesaler by Adolf Würth in Künzelsau. Today, the company, with a history of more than 75 years, sells over 125,000 items. With over 7571 employees in 2022, AWKG achieved sales of EUR 2.5 billion. The company's 3,244 sales representatives serve more than 660,000 customers across Germany. As the largest single company within the Würth Group, AWKG is the parent company of the globally active group. With sales of EUR 19.95 billion, it is one of the world's market leaders in assembly and fastening materials. With

the principle “Every customer his Würth“, the company focuses on its customers and strives to offer each customer individually designed systems and services in addition to its broad product range. One of the best-known system solutions is the ORSY shelf. This offers every customer the opportunity to individually design a storage system tailored to their own work processes. The Board of Directors of Adolf Würth GmbH & Co. KG consists of eight members with different areas of responsibility. These are divided into the areas of sales, sales force, finance, logistics, sales management, and internal sales, among others. As part of the Würth Group, AWKG is subject to the directives of the group. The group is managed by business area heads, who look after the strategic business units. The business area managers are in turn subject to the five-member group management board. The Advisory Board is the highest supervisory body. It deals with matters relating to approvals for corporate planning and questions concerning strategy development.

1.3.2 Presentation of the sales structure of AWKG

AWKG’s distribution model is characterized by a strategy of multi-channelism. As can be seen in Figure 1.1, this model can be represented in the form of a cube with the three dimensions of customer contact point, sales channel, and division. All customers, as well as salespeople for the company, are assigned to one division and one sales channel, respectively. The choice of the customer contact point is the responsibility of the customer. Multiple customer contact points can also be used in parallel.

The sales channel dimension can be divided into key accounts, regional sales, and telephone sales. In telephone sales, the sales staff works in the office and is responsible for the smallest customers. These are customers with little sales potential. The aim is to look after these customers on a regular basis but at a reasonable cost. Key account customers include customers who have large sales potential and usually have multiple locations. The sales force is responsible for negotiating framework agreements for all locations. Regional sales represent AWKG’s field sales force. All other customers of the company are served through this channel. Customer contact points are the marketing channels through which contact is established between the customer and the company. Each customer is served by a salesperson based on their assigned sales channel. In addition, there is the option of establishing contact via telephone calls or e-business. These include the online store and the app. In addition, a purchase can be made directly via a Würth branch. In Germany, customers can shop at over 550 locations. The final dimension of the sales model is the division. Auto, construction, metal, and wood represent the four divisions. These, in turn, can be broken down into divisions, each of which contains a distribution channel of the same name.

1.4 Pilot Interview

This thesis is based on building a visualization tool for the business that helps to find the interaction and relationship between the customer and the product. The pilot interview is conducted as per certain guidelines provided by the authors of [SMM12] and [MD20]. As the authors of [SMM12] mentioned, we have to analyze a real world problem faced by business (domain experts), then build a visualization system that supports solving that problem, validate the design of the system, and refine the visualization system based on the inputs from the validation. The visualization study

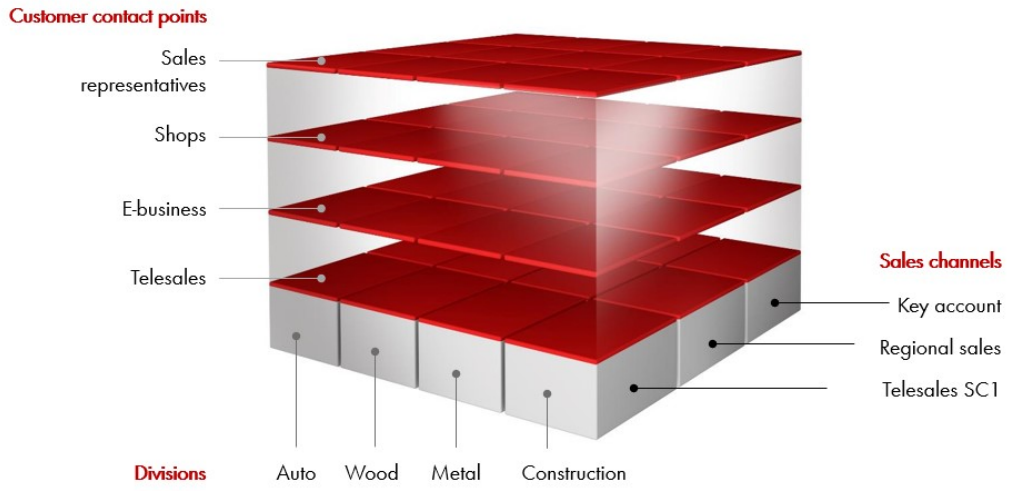

Figure 1.1: Multi-channel approach in sales at Adolf Würth GmbH & Co. KG (source: internal document)

suggested by the authors of [SMM12] is based on three phases further classified as a nine-stage framework, namely, the Precondition phase (learn, winnow, cast), the core phase (discover, design, implement, and deploy), and the analysis phase (reflect, write).

Based on the problem, the learning phase will be further developed in the Chapter 3 with the related paper works. In the winnow phase, we select the candidates for proceeding with the visualization tool. This plays a crucial role, as it paves the way for how we conceive the problem and develop it further. Also, to plan the timeline for the project delivery, analyze the real need and decide the target audience and effectiveness of the tool, As with the winnow phase, I conducted a pilot interview with two technical people from the data science team and one non-technical person, the manager of the data science team. Both groups are approached with the same questions. The questions are related to the recommendation model that was built for the purpose of product bundling. So, with this as the basis, I framed the questions that they expect from the machine/deep learning-based model's performance and the need for results interpretability. But this came up after explaining the model developed to aid the case of product bundling, which lacked post-hoc explanations. Based on the interview, the visualization tool was built. I have summarized the outcome of the pilot interview in the following points: Here, we consider group 1 to be the two technical people from the data science team, and group 2 represents the non-technical manager of the data science team.

- This is the main input that came from the business point of view, i.e., group 2. As part of using the recommender system, the main challenges are explaining the results to the sales representative for the reason why the customer is being suggested the products and focusing mainly on the products that the customer has already purchased so that the results aren't suggested without relevant products for the customer, which causes dissatisfaction for the customer.

- This was the suggestion from both groups since both had problems providing the solution with an explanation specific to a particular group. The information typically needed for making or taking action is based on how the results are broken down and explained. For this purpose, it is suggested to have two views: the internal user view and the customer/non-technical person view, so that it is easier for the end user to understand.
- Group 2 requested a visualization that provides a comprehensive view of the products in their hierarchical structure, enabling them to comprehend the representation of recommended products and the ones already purchased by the customer. Also, with the help of visualization, we can provide KPI representations of the purchased products to find the most purchased products on each product family tree level.
- Group 1 with the technical team decided on the timeline and usability of the tool. The tool must be provided with documentation and help text to be used. Currently, deployment is planned to be local and, in the future, scaled to specific user groups.
- Group 2 suggested the theme be followed with the right color mapping, and the logo must be placed.
- Understand the existing data and plan to deliver the tool within a timeline of two months as agreed with the domain experts. Here, the domain experts are from both groups.

From these inputs, it becomes clearer that Group 1 requires more insights into how the recommender system functions, aiming to unveil the workings of the black-box model and understand its learning patterns. Conversely, Group 2 emphasizes the importance of business value, focusing on satisfying customers based on decisions made by the machine learning model to maintain their satisfaction. Another aspect is understanding existing data through visualizations. Considering these inputs, the tool will be developed within the given timeline. Subsequently, a user interview will be conducted to evaluate the tool's effectiveness and usability. In the cast phase, it is important to differentiate the end users of the tool, the project approver (gatekeeper), and the translators who can communicate domain problems in a more generic manner to non-experts. The discovery stage involves requirement analysis in software engineering, followed by the explanation of design and implementation stages, selecting a specific method, and testing with end users in Chapter 5.

1.5 Outline

The structure of this thesis is outlined in the following manner: It is comprised of eight chapters, arranged in the following order:

Chapter 2 introduces the concept of artificial intelligence and its subsets like machine learning, deep learning, and reinforcement learning, and also discusses recommendation systems and data mining concepts.

Chapter 3 deals with the technological support for the chosen methods by having a literature review on the topics discussed.

Chapter 4 provides an overview of the data infrastructure and the data that's getting used for the project, as well as methodological concepts for solving different approaches like neural collaborative filtering, the alternate least squares method, and frequent item mining.

Chapter 5 discusses the result comparison of the methods discussed in the previous chapter and benchmarking them. Also, the technical details for the visualization toolbox.

Chapter 6 explains how the results are evaluated and how to derive insights from them, and also discusses the outcome of the visualization toolbox as part of the survey.

Chapter 7 provides insights about the future scope and summarization of the outcome of the thesis work.

2 Background

This chapter deals with the technical background details required for understanding machine learning, deep learning, and recommendation systems.

2.1 Artificial Intelligence(AI)

Previously, humans made decisions based on data that was easily interpretable and of a meager amount. But the data started to grow tremendously even from the small level of data collection; for instance, sensor data that gives live data for autonomous systems, sensor data from medical devices, data from web scraping, and payment systems from all over the world and this leads to big data, and handling the data to make decisions becomes problematic and really time-consuming. As mentioned in [TUR50], Alan Turing's Turing test paved the way for artificial intelligence. He conducted tests with an interrogator with humans and machines to find the difference in response between humans and machines. Through this test, he framed the question "Can machines think like humans?". During this extensive research, some events occurred that either supported or contradicted the question. So, to explore this question, researchers have made good progress in making machines learn like humans and have succeeded in many fields like medicine, business, education, etc. in making data-driven decisions and solving complex tasks. Now many decisions depend on the decisions made by AI-powered machines, which, on the other hand, could take longer for humans to get those results given the complex data structure. So, AI is the ability of machines to solve complex problems with the help of infrastructure and robust datasets. As given in figure 2.1, AI has two major subdomains, namely, deep learning and machine learning. This will be explained clearly in the following sections.

2.1.1 Machine Learning

Machine learning is a subfield of artificial intelligence (AI). Generally, intelligence refers to the ability to learn, and in the context of evaluating a machine's learning capability, it is referred to as artificial intelligence. In machine learning, predictive models are built to solve business problems by learning from data. This detects patterns by learning the data and provides results based on the learned pattern. With less human involvement, machine learning can analyze, understand, and detect patterns for making decisions. The main components of machine learning are the datasets, features, and algorithms. Machine learning works on both structured and unstructured datasets. Feature learning is for the weightage of the features that constitute the prediction, and algorithms are based on the problem, data, and accuracy. Sometimes, multi-modal learning (ensemble learning) is done for accuracy and performance. There are three different learning methods based on the data provided to the model:

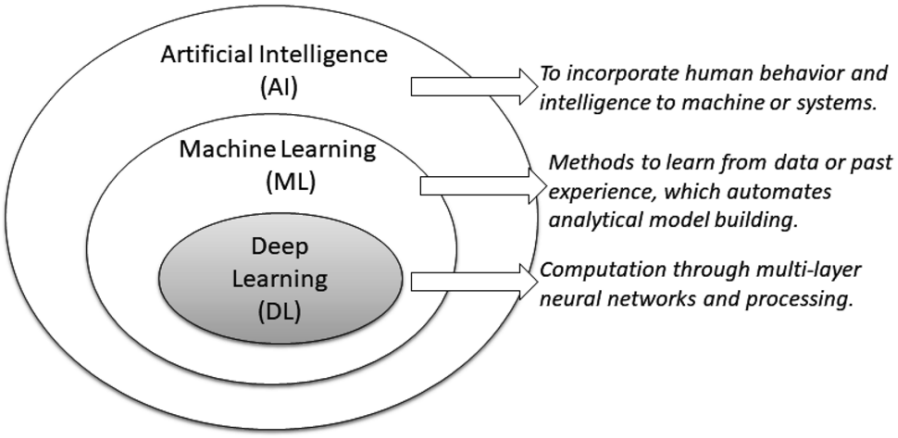

Figure 2.1: An illustration of the position of deep learning (DL), comparing with machine learning (ML) and artificial intelligence (AI) [Sar21].

- **Supervised Learning:** If the data is labeled, i.e., providing information about the data as input, the model can learn the labeled data and predict the data based on the learned pattern. The model's performance is measured with the actual labeled data. A few of the supervised learning algorithms include logistic regression, decision trees, SVM, and random forest. Supervised learning is further divided into:
  - **Classification:** The model learns to discover the group to which the discrete values belong and separate them into several categorical classes. It will predict the probabilities into multiple classes with the decision boundary, and the classes with the highest probability will fall into that class. The performance is measured with the classification matrix, Receiver operating characteristic (ROC), and Area Under Curve (AUC). Example: Classification of mail as spam or not.
  - **Regression:** The model here learns with continuous values, where the model maps the predictive relationship between labels and data points. The performance is measured with mean square error (MSE) and root mean square error (RMSE) values because the model tries to approximate the mapping function from input variables to continuous output variables. Example: sales forecast, weather forecast.
- **Unsupervised Learning:** This type of model uses unlabeled data to discover patterns in the data and identify hidden features in the training data. Clustering approaches like K-means, DB scan clustering, and Gaussian mixture models are the most popular techniques in this category. Example: customer classification and churn.
- **Reinforcement Learning:** As defined in [KLM96], reinforcement learning is a method to sequentially self-correct from environmental feedback (positive or negative) and therefore improve the overall model function without having labeled data. It works like a reward-based model: the agent tries to achieve a reward and avoid punishment, which provides the best outcome for each action.

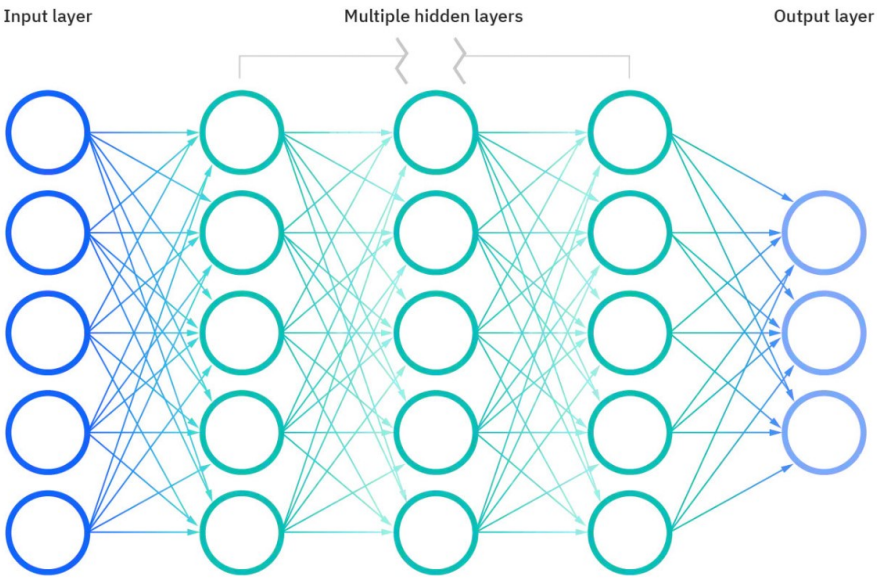

Figure 2.2: An illustration of the artificial Neural network with input, hidden, and output layers [JIA21].

2.1.2 Deep Learning

Deep learning is a subset of machine learning where the learning happens through neuron structure, mimicking the way that biological neurons signal to each other. It can process vast amounts of data. Deep learning's core concept lies in artificial neural networks, which enable machines to make decisions. Deep learning deals with deep neural networks. Neural networks learn based on their weights and biases. Based on the number of hidden layers, the total layer architecture differs. As in figure 2.2, the number of neurons in each layer will be connected to the previous layer's neurons and taken as input. Each neuron is connected with a weighted edge. The sum of weights is passed as input to the activation function, which decides whether the neuron should be fired or not, and then it gets passed to the output layer. Weights denote how important the input value is.

In neural networks, each neuron takes the weighted sum of its input values, which are multiplied by corresponding weights. Additionally, a bias term is added to the weighted sum, Eq. (2.1). As an instance, if the inputs and weights are as Eq. (2.2) & (2.3), respectively, then the weighted sum and bias(constant) are computed as Eq.(2.4). Bias is used to shift the result of the activation function towards the positive or negative side. This determines the control of the activation function. The activation function normalizes the inputs as in Eq. (2.5).

$$Y = \sum(weigth * input) + bias \tag{2.1}$$

$$inputs = x_1, x_2, x_3, .., x_n \tag{2.2}$$

$$weights = w_1, w_2, w_3, .., w_n \tag{2.3}$$

$$Y = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + .. + x_n * w_n + bias \tag{2.4}$$

$$output = activationfunction(x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + .. + x_n * w_n + bias) \tag{2.5}$$

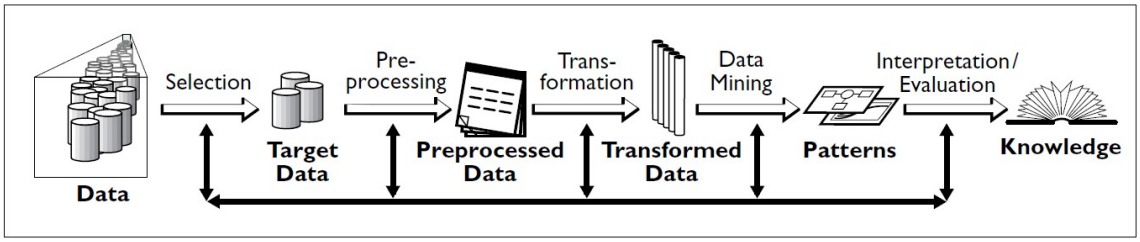

Figure 2.3: Overview of the steps constituting the KDD process [FPS96].

If the models' output is not performing better as compared to the actual output, the model uses the backpropagation technique to improve the performance of the network. Based on the error rate (loss) obtained in the previous epoch of training, the weights are fine-tuned to get lower error rates. This makes the model reliable by increasing its generalization capability. This makes the model robust for deep learning tasks. Depending on the requirements, the number of layers and hyperparameters are adjusted for better accuracy and learning. There are different deep learning techniques based on the provided inputs and needs. For instance, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and General Adversarial Networks (GAN) are famous deep learning techniques prevalent in usage and used according to their purpose in real life.

2.2 Data Mining

The amount of data generated in the modern era is enormous. The research was more focused on getting insights from this type of huge data stored in databases, which helps in making data-driven decisions, which led to one of the techniques as defined in [BR10], known as data mining. Data mining is the process of extracting useful information and data patterns from large amounts of data. It is also known as the knowledge discovery process (KDD).

As in figure 2.3, data mining is the important step to derive the useful pattern from the data, but in order to ensure the data is of good quality to be interpreted, we need to follow the data preparation steps that are provided below,

- **Domain knowledge:** By acquiring a deep understanding of the business domain, we can effectively define and align the application's objectives, ensuring they address the specific needs and requirements of the industry.
- **Target dataset:** Filtering out the necessary dataset or feature variables for which the knowledge discovery is to be made.
- **Data cleaning and preprocessing:** Once data is filtered, it must be prepared to be without outliers by sampling the data for matching the data imbalance, handling the missing data fields that would be necessary, or else removing them by using statistical analysis, mapping missing and unknown values, and addressing schema and data type issues in DBMS-related parts.

- **Transformed Data:** After cleaning the data, we need to transform it according to the choice of the algorithm and the problem. By employing dimensionality reduction methods for reducing the effective number of variables under consideration or identifying the invariant representations for the given data.
- **Data Mining:** After data preparation, choosing the specific data mining algorithm for the purpose of knowledge discovery and the parameters that are appropriate and according to the user's preference over the results. The high-level goals of data mining are prediction and description. **Prediction** is to find the unknown values of dependent variables based on the independent variables or values in the database. **Description** focuses on interpretable patterns based on the data. User preferences are more about the interpretation than the prediction itself. These high-level goals are achieved by the following data mining algorithms:
  - **Classification:** Basically, to learn a function that classifies a data item into any one of the several predefined classes, Famously known as the C4.5 classification algorithm, it generates a decision tree for the purpose of classification and uses information gain as splitting criteria. For the continuous values input, based on the threshold values, it divides the value above to be one part and below or equal to another part.
  - **K-means:** It comes under unsupervised learning (i.e., unlabeled data), and the purpose of this algorithm is to group or cluster the data points of similar patterns. K-means is based on the "centroids" of each data point with K-group. Later, it assigns each data point to a cluster based on its closest centroid. From clustering, we could find broad distribution patterns and relationships across data variables, as well as dense and sparse areas in object space.
  - **Support Vector Machines (SVM):** It is also a classifier algorithm, but the line of separation is only between two groups. Support Vector Machines take data points and project them in higher dimensions. With this, it can more accurately split the data points into two groups by separating the hyperplane.
  - **Apriori:** The Apriori algorithm is useful for extracting the data points that are more closely associated with each other and can be formed as a group with the associated data points from the dataset. It is also called frequent itemset mining. This method is helpful in realizing the product bundle recommendation, which will be dealt with more in detail in the chapter 4.3.
  - **Expectation-Maximization (EM):** It is a clustering algorithm based on statistical inference. This algorithm aims to represent the points in the dataset in the graph and clusters the points to be in the bell curve to form groups.
  - **PageRank:** PageRank is the most influential algorithm; it ranks the web search results based on their relevancy. The ranking is based on the relevance of the webpage to the query made and the corresponding linking web pages. If the scores are said to be high, they will be ranked at the top of the list.
  - **AdaBoost:** It is a supervised classifier method that compares the results from other classifiers and provides better results if the classifier's performance is not good. This work takes an iterative approach to each misclassified data point and works on it to provide the best at each iteration. This forms a complex decision tree classifier.

  - **K-Nearest Neighbours (KNN):** It is a lazy classifier as compared to other classifiers, as they are eager learners. A lazy classifier works when a new data point is placed; it calculates the distance of its neighbors, and the more relevant the data point, the more likely it will be assigned to that class.
  - **Naive Bayes:** This classifier makes the assumption that each feature in the set of data is uncorrelated with all other features. Based on this assumption, this algorithm predicts classes of unseen data.
  - **CART:** It is a combination of classification and regression in a tree-based algorithm. It's based on the binary tree; each node will have exactly two edges, which makes it less susceptible to noisy data where the outliers are higher. This is a robust classifier based on decision tree classification.

- **Data Mining Applications:** Data Mining has been widely used in retail stores, hospitals, banks, and financial companies. These companies reap the benefits of data mining by finding patterns, rules, and classifications in their data. This helps with the data-driven decision-making purposes, which would have taken longer based on the manual mining. Applications include those in Insurance companies for detecting fraud transactions, increasing the efficiency of marketing campaigns, cross-selling based on the customer's group, the policy approval process, etc. Data mining plays an important role in business decisions for the discovery of knowledge or patterns by using different algorithms according to the business problems. Data mining is viewed as a significant area in database and information systems and an interdisciplinary advancement with great potential in the field of Information Technology.
- **Data Mining Limitations:** Even though, as mentioned in [FPS96], data mining is effective for finding patterns and for knowledge discovery on data, It has certain limitations.
  - **Size:** If there are databases with millions of records and a large number of features, these create large exploration search spaces, which allow the data mining application to find patterns that are illogical and invalid.
  - **Redundant pattern:** Sometimes the patterns that are generated won't be helpful because of their understandability. The rules generated cannot be understood by humans to evaluate the results.
  - **Dataset issues:** If the dataset being used keeps changing, the derived pattern will be invalid. Features that get updated or changed will also cause the pattern to be mismatched.
  - **Non-standard data & integration:** Currently, KDD algorithms won't work with data that is not numerical. Non-standard data types are stored in the database now as part of the data lakehouse, which makes the KDD process not viable. Integration of the results with the DBMS is often cumbersome and requires a lot of computations for the integration process.

2.3 Recommendation Systems

On an e-commerce website or online news site, there are multiple items for the customer. As for a single customer, it will be more useful if they are able to view the contents of their interests instead of hoarding all the items available, which would reduce the customer's interest in the website or product that is overwhelming them. So, suggesting the content or product as per their interest would be more important for users and business organizations. So, an algorithm that evaluates user data and offers individualized recommendations for products or actions is referred to as a recommendation system, also known as a recommender system. Based on a user's past behavior, preferences, or demographic data, these algorithms are frequently used in e-commerce, social networking, and content platforms to recommend products, services, or content they would find interesting. By promoting products to users that are more likely to be bought or consumed, recommendation systems aim to enhance the user experience, boost user engagement, and ultimately drive revenue for the organization. Also, to make the user satisfied with the recommendations, which encourages them to purchase or visit again, a recommender system aims to decrease information overload by recovering the most relevant information and services from a large amount of data and providing individualized services. For example, in the use case mentioned in [GH16], data from Netflix's site shows that 80% of what people watch comes from their recommendation algorithm. As a result, recommender systems generate over $1 billion per year of Netflix's revenue.

2.3.1 Recommender methods:

The recommendation system is calculated based on the user's or item's characteristics or purchasing history. As in figure 2.4, the recommender systems are mainly categorized into content-based filtering, collaborative filtering, and hybrid filtering techniques. This will be discussed in detail below.

Content-based filtering: Based on a single user's activities and data, the recommendation is made to the user. It depends on the description of the item that the user is interacting with and the user's profile. The application of content-based filtering would be for single users preferences and usage, like web pages and news articles, where the content plays a more important role in deciding the recommendation. The steps that are followed for content-based filtering are first to detect the item descriptions that are preferred by the user and get stored in their profile. Then compare the general item descriptions and each item's attributes to filter out the items that are getting matched with the user profile's item descriptions. Finally, only those items will be displayed to the user as per their preference. But this approach has problems when the data is scaled, there won't be much diversity in the results, and it is not always clear which aspects of the product the user likes or dislikes.

Collaborative filtering: In collaborative filtering techniques, recommendations are made based on the similarity of the user along with other users interactions or behaviors. This is the most extensively used approach in many domains as a recommendation system because of the diverse and accurate recommendations it makes. By comparing the preferences and interests of other users who have rated things that are similar to those that the active user has rated, recommendations are made to that user by this technique. As per the author [BGB15], collaborative filtering considers more than one feature to recommend items to the user or user groups. Generally, the dataset must

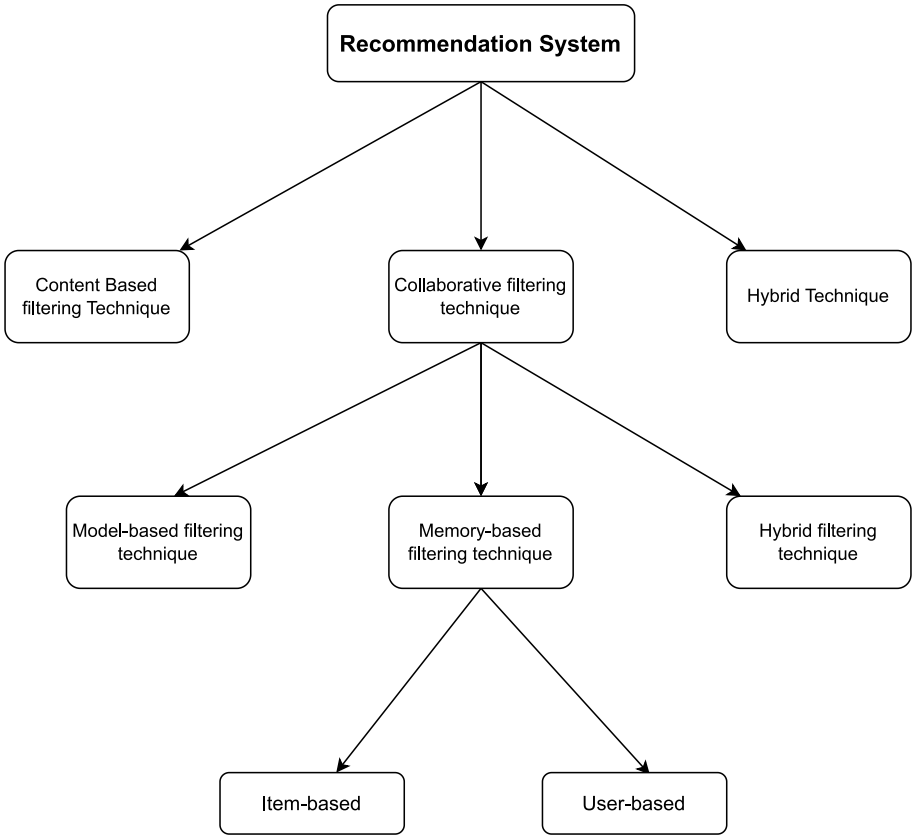

Figure 2.4: Classification of the recommender Systems [IFO15].

have a large sample size to know the exact interactions between the user and items. As in figure 2.4, collaborative filtering is classified into three types: model-based filtering, memory-based filtering, and hybrid filtering techniques.

(i). Memory-based filtering Technique: With the inputs from the authors of [Cac11], the algorithm can find a user's neighbor who has similar interactions by looking at products that have already been purchased by that user. This is basically done by using the similarity measurements between the user and user group with the similar products being purchased. User-based and item-based filtering techniques are the two main categories of memory-based filtering techniques.

- **User-based filtering:** The user-based filtering technique is based on the similarity measures between the users, not the items. It works by finding the similarity between users by comparing their ratings or interactions on the same item and then predicting the rating for that item. As mentioned in [IFO15], the recommendation score is calculated based on the predicted rating and provides the recommendation of items according to the similarity between the users.
- **Item-based filtering:** The item-based filtering technique computes recommendations using the similarity between items and not between users. From the user-item interaction matrix, the similarity of the targeted item to retrieved items will be computed, and then the item will be ranked with a similarity score. As described in [IFO15], based on the top-most similar items, the products will be recommended to the users.

(ii). Model-based filtering technique:A model-based filtering technique is used to generate the model from the dataset and provide recommendations to the user. It is based on the prior rating of the users. The model learns based on the pattern, and it will be added on top of collaborative filtering for the edge cases. Pattern mining is based on KDD (Section 2.2) or machine learning techniques, for example, singular value decomposition, dimensionality reduction technique, clustering, decision tree, and association rule mining. It resolves the sparsity problems associated with recommendation systems.

- **Association rule:** Based on the user and item, it provides recommendations as rules. The rules are extracted as follows: if two products are bought together, there is a high chance of getting the third product together based on the user's purchase history.
- **Regression:** Regression based models are built to find the linear relationship between provided inputs or independent variables and the outputs that are produced.
- **Decision tree:** A more interpretable method for providing recommendations gets trained on labeled data from training examples and then applied to unknown data examples.
- **Bayesian classifier:** This technique is based on the definition of conditional probability and the Bayes theorem. It is considered to be a random variable. The most common Bayesian classifier is the naive Bayes classifier.
- **Matrix technique:** Based on the user-item interaction matrix, this method computes the unknown relationship between user-item based on matrix factorization techniques.

(iii). Hybrid filtering technique: Hybrid filtering technique is the recommendation method, which is a combination of two different recommendation methods implemented separately and then combined for the output. As for the optimization provided by the authors in [IFO15], model-based techniques will be combined with memory-based techniques for system optimization and to overcome the limitations of certain methods. But on the negative side, the computation will be heavy.

2.3.2 Recommender Systems Challenges:

As described by [KAU16], the following are the challenges that are found in recommender systems,

- **Changing user preferences:** The recommendation system is mainly based on the user's interests and profile. But these continuously change over time.
- **Sparsity:** When only a limited number of interactions or ratings are performed on an item, the number of available items is reduced. If the recommendation system works based on the similarity of the neighbor and that neighbor's data is sparse, then the recommendation algorithm won't be performing well. This is called a **Cold start problem**.
- **Scalability:** With the increased number of users and items, the applicability of recommendation systems will be difficult because the algorithm will need more resources to run.

- **Privacy:** If recommendation systems are provided with a number of attributes (for example, demographic data, and location), that will ease the recommendation process. But due to privacy, data can't be used, which limits the performance of the recommendation systems.
- **Synonymy:** It is the likelihood problem with similar items. For recommender systems, it will be hard to distinguish between the related items and make a suggestion to the user. For example, screws with different dimensions.

3 Related Works

The related works chapter deals with the novel contributions provided by authors regarding the methodologies of recommendation systems, product bundling approaches, and visualization techniques.

As demonstrated by Jawaheer et al. in [JSK10], the recommendation system is generally built up based on implicit or explicit feedback. Implicit feedback is typically passive and requires no additional effort from the user's perspective. For instance, the click-through rate indicates whether the user clicked on a particular item or link, how long the user spends viewing items based on the user's past purchasing history, search queries, and the keywords that the user entered when performing a search. We can infer it as a parameter to measure user behavior, preferences, and interests. On the other hand, explicit feedback is based on user ratings and reviews of a product or item. It directly measures the real interest in the product, but the explicit feedback won't be intentionally provided by the user. As shown in [JSK10], those data would be mostly sparse. As the focus of the product bundling recommendation is based on implicit feedback, in our case, the rating or reviews aren't available for the whole customer base.

Beel [Jör17] suggests the content-based filtering algorithm is based on the tags and the names of the items that each user has already purchased and provides a recommendation to the user with additional items that are of similar properties. Even though it's a personalized recommendation (as the recommendation is done per user), it has certain drawbacks, like the fact that the computing time will be longer and that it will also recommend the user with similar items that they already know of. With the content-based filtering technique having its drawbacks, the next type of recommendation system is Collaborative Filtering(CF) technique.

Goldberg et al. [GNOT92] introduced collaborative filtering for the purpose of filtering emails as good or bad or associating text annotations with the emails. Collaborative filtering works based on the feedback provided for an item by a user, and with this as input, it finds the relationship between the user and the item and then recommends new items to the users. According to Bokde et al [BGM15] approaches that existed for Collaborative filtering were not scalable due to the fact that the information about the product is growing at an exponential rate. This demands a more efficient implementation, where the previous method dealt with association inferences, which were time- and resource-intensive. For this, they suggested an efficient and scalable matrix operation, Matrix factorization (MF). Based on the rating provided or created for each user and item, the Matrix factorization (MF) represents the user and item as vectors to represent the relationship between them and find the correlation. This recommendation method using matrix factorization provides accurate results and is highly scalable.

Based on the matrix factorization, item-based and user-based collaborative filtering methods are computed. In the user-based collaborative filtering method [HKR02], the user rating is considered for the recommendation with the help of active user neighborhood information. This calculates the neighbors of the user with the user considered, and based on that, the user will be recommended items

from the neighborhood users. In this paper, Sarwar et al. [SKKR01], discuss the benefits of using item-based collaborative filtering. The authors investigate finding the user-based recommendation with item-item similarities using two methods: item-item correlation and cosine similarities between item vectors, then compare the results for finding the recommendation to the users. The cosine similarity is calculated by considering the items in user-dimensional space, and then the similarity between the items is calculated by computing the cosine of the angle between the vectors. The correlated-based similarity is calculated based on the Pearson coefficient, and the adjusted cosine similarity overcomes the drawback of the cosine similarity by considering a different rating scale for all users by subtracting the corresponding user average from each co-rated user pair.

Though the results of collaborative filtering are feasible, they have disadvantages such as the cold start problem, scalability, and popularity bias. Takács et al. [TT12] mentioned that the problem with collaborative filtering is rooted in the sparse matrix of customer and product interaction being empty, which leads to the large sparse matrix. So, by decomposing the customer-product interaction matrix into two lower-dimensional rectangular matrices, One of the matrices is the user matrix, in which rows represent the users and columns represent the latent factors, and the other is the item matrix, in which rows represent the latent factors and columns represent the items. Latent factors are used to represent the features projected from the customer-product interaction matrix. By using the matrix factorization and the tuning of the regularization parameter (L2), Alternate Least Square processes the customer-product interaction matrix like it keeps the customer matrix fixed and runs gradient descent with the product matrix; then it keeps the product matrix fixed and runs gradient descent with the customer matrix.

Even though the results of matrix factorization with ALS are better than those of traditional methods, the recommendation could be improved by learning the non-linear features instead of only latent vector multiplication as done with matrix factorization. Salakhutdinov et al. [SMH07] defined two-layer Restricted Boltzmann Machines (RBMs) to model users' explicit ratings on items, which introduced the usage of a neural network-based recommended system that was used to learn the ratings. This was extended to have an ordinal relationship between the rankings.

Similar to the item-item recommendation model, Sedhain et al. [SMSX15] claimed that the usage of auto-encoder models generalizes the rating of the user and reconstructs the ratings of them by the decoder part. But the problem with this model is the difficulty in generalizing the unseen data and reconstructing the recommendation. To avoid this, Li et al. [LKF15] proposed that with the help of denoising autoencoders (DAEs), the inputs are corrupted and then fed to the model. So, that model learns to process the unseen data and provides recommendations.

He et al. [HLZ+17] examined the added advantage of including the neural network architecture in addition to the existing matrix factorization method. With the help of a multilayer perceptron, the model learns non-linear features to get more interactions than linear feature learning by matrix factorization. The authors compare the two datasets with general matrix factorization, multilayer perceptron, and combinations of both. In comparison, the combination of General Matrix factorization and multilayer perceptron performed well, supporting the claim of non-linear learning through multilayer perceptron. This helped in deciding the method that will be applied to this recommendation system.

As of now, all results are based on recommendation systems. For the purpose of checking the feasibility of finding the most frequently bought products, we use frequent itemset mining. Sharma et al. [SB17] proposed the association rule mining method, which is a data mining method to retrieve

the rules from the databases that are frequently occurring or bought together. With the provided transactions of different items, this method discovers the rules that determine the connectivity or relationship of the items present. Further, the authors evaluate different frequent itemset mining methods and discuss their advantages and disadvantages. With the help of the FP-growth algorithm, the frequent items are derived without the candidate generation. It has the advantage of consuming less space, but it ends up mining only generally occurring rules and also generates a huge number of rules that might not be frequent as a whole. So, from this, we can understand that the use of Associative rule mining will be better for our case. Even FP growth provides an advantage in resource consumption, but Associative rule mining produces good results.

For conducting the visualization of the sales relationships and interpreting the existing data, we need a proper visualization study to be conducted. Sedlmair et al. [SMM12] demonstrated a nine-stage framework for conducting a visualization design study to properly design a visualization system that solves the problem, evaluate the design, and refine the visualization by reiterating the lessons learned. It is combined with an explanation of pitfalls due to the common errors that will result in the failure of the visualization system design. With the help of this approach, we could build the visualization system efficiently, ensuring quality and efficacy.

Meyer and Dykes [MD20] created a framework consisting of six criteria to ensure rigor in design studies. Those criteria are, informed, reflexive, abundant, plausible, resonant, and transparent. The researchers approached the development of these criteria using a deductive and top-down approach, drawing upon established criteria and principles from the field of social science. As a result, the criteria they developed are broad and encompassing, providing a high-level perspective on rigor in design studies.

As for interpreting the recommendation model to understand each hidden layer's functionality and the results, we need a visualization toolbox to visualize the interactions so that the model's learning can be understood. Ricci et al. [RRS11] explains the trade-off between the model's explainability and accuracy. If the model is simple and accuracy is at stake but it can be explained easily, it would be difficult with a complex model with good accuracy but no explainability. So, choosing a model with both good accuracy and explainability will be a trade-off.

Explainable recommendation models are generally either model-intrinsic or model-agnostic. The model intrinsic approach [ZLZ+14], provides interpretable models that aid in making the decision mechanisms transparent and explainable. On the other hand, the model-agnostic approach [WCY+18],is the post-hoc explanation method in which the model remains a black box and provides pattern learning from the outputs of the provided model.

Zhang et al. [ZC18] discussed various explainable recommendation models and conducted a survey about the existing methods. Explanation can be of different forms like displaying relevant user or item explanation, feature-based explanation, opinion-based explanation, sentence explanation, visual explanation, and social explanation.

Resnick et al. [RIS+94] and Cleger-Tamayo et al. [CFH12] provided a plain explanation about the relevant user or item by considering the user's neighbors who rated items similar to the user. It provides a histogram for suggesting the user with the closest possible user groups and provides the item that falls into the group to the user considered.

Vig et al. [VSR09] conducted an experiment on feature-based explanations with content-based recommendations by taking movie tags as features for the purpose of recommendation and explanation. The basic idea is to find the tags from the user-liked movie and then recommend items based on the collected information that gets matched, and the features that match will be displayed as relevant tags. Ren et al. [RLL+17] proposed a social collaborative viewpoint regression (sCVR) model that gives explanations as viewpoints. Viewpoints are the sentiment labels of the user's social circle. This gives a better understanding of the user's circle of friends liking and why the items are suggested to the user, as well as generalized findings for the opinion-based explanations.

Zhang et al. [ZLZ+14] provided an explanation with text based on the template, like the user might be interested in a certain feature, and based on that, the new products are being suggested, referred to as sentence-based explanations. This was also extended to provide a dis-recommendation like the user doesn't like these features, so the new products aren't being suggested.

Lin et al. [LRC+18] demonstrated a method for visual-based explanations with the help of a convolutional neural network with a mutual attention mechanism to extract the features available in the clothing dataset and feed them to the model to make predictions of the rating scores of the recommendations. This will provide a visual explanation, like which part of the visual features is contributing more for prediction purposes.

Park et al. [PJK+17] focused mainly on eradicating the problem of personalization since the users are recommended to generic user groups of similar likings, which doesn't mean that the user is also interested in their likings. So, if the user is getting suggestions from their friends or social group, the user will be satisfied with the suggestions. So, the authors proposed a graph-based visual explanation that considered ratings and the social information of the user to generate explainable product recommendations.

The main problem of [RRS11] with the explanation of the recommender systems that deal with matrix factorization is the latent factor representation of the user and item vectors, because with the latent factor we cannot derive the exact factor that derived the predictions or recommendations. So, Zhang et al. [ZLZ+14] discovered Explicit factor models (EFM), which take the user's favorite features and train on them so that the recommendation is only based on that feature. Finally, it can be explained by the fact that the user is recommended the product because of the user's favorite feature and that the feature contributes more importance to the recommended product.

Bauman et al. [BLT17], added their contribution by including the review-based explanations for the user. They proposed the Sentiment Utility Logistic Model (SLUM), which extracts the features and sentiments as input to the matrix factorization model to find the ratings for the new recommendations. This method provides not only product recommendations but also feature recommendations for each item as explanations.

Tao et al. [TJWW19] proposed to integrate latent factor models with factorization trees. The authors used regression trees on users and items with user-provided reviews, and based on those reviews, the latent profile for each node on the trees was made to represent the user and items. By doing so, it is possible to track the creation of latent profiles by tracing back the path of each child to the source parent node, which serves as an explanation for the item that is being recommended to the user.

For the explainability of the latent factors with a model based approach, Abdollahi et al. [AN17] examined a method using an explainability regularizer into the objective function of the matrix factorization. If many of the user's neighbors also bought the item, the explainability regularizer

makes the user and item latent vectors close to one another. By doing this, the model suggests the items that fall under the neighbor's circle, giving the explanation that many similar users have purchased this item, and so it is recommended for the user.

Since the model complexity increases, explainability reduces, as mentioned in the tradeoff between accuracy and explainability. So, the post hoc explainability model research was started. For this, Peake et al. [PW18] suggested providing a post hoc explanation with the help of association rules. With the model's input applied to association rule mining, the rules are mined, and if the results from the model that is used for recommendation produce an output with the item that is getting matched with association rules, then a simple explanation is made with the rules, like product A is bought, so product B and C are bought together, which is defined with the help of confidence and lift parameters.

Further, Marco et al. [RSG16] claimed LIME (Local Interpretable Model-Agnostic Explanation), which uses sparse linear models to simulate a non-linear classifier around a sample, allowing the linear model to reveal the feature(s) of the sample that was responsible for the label that was predicted. This can provide local explainability for a single sample.

With the base of game theory, Scott et al. [LL17], established SHAP (Shapley Additive exPlanations) which explains the output of any machine learning model being model agnostic. Based on the Shapley values, it calculates the local gradient of each local value, counts the contribution of the features for each, and provides which features are contributing towards the end results.

Through all these works, we can convey that considering post hoc explanations would be better at explaining the model. Since the processing of deep learning with hidden states provides the learning of the pattern, it can't convey the strong reason with attributes that contribute to the output. Since the hidden layers are converged and concatenated with machine learning output, This strongly supports the fact of post hoc explainability. In this work, we will discuss how the learning of the model happens at the final hidden layers by computing the output from that layer. And by using dimensionality reduction techniques, we will reduce the lower-dimensional representation and then compare the output from each layer with the concatenated layer to find the pattern, which we can convey based on the group of customers to whom the products are recommended. Further, we will check the activation functions of each hidden layer to see which features contributed most to each hidden layer with the help of heatmaps and also check for the hyperparameter values that influence the model's performance.

4 Methods

4.1 Data Lake as Information Basis

4.1.1 Presentation of the Data Lakes

In the era of Big Data, the importance of using more and more diverse data sources, which are to be stored in a consolidated manner within a system and subsequently made analyzable, is constantly increasing. In this context, data lakes are becoming increasingly interesting as data storage facilities. A data lake is a concept that enables the storage of large amounts of data based on many cost-effective technologies. Thus, a data lake can serve as a company-wide data management platform and enable the analysis of data from different sources. It is also possible to store barely processed data and raw data in their original form. Since homogeneity between the individual data does not have to be guaranteed, data can usually be stored as files, originating from a wide variety of sources. Structured data, as in relational databases, as well as semi-structured data, e.g., in the form of CSV files, can be stored on the data lake. In addition, it is possible to store images, videos, or e-mails as unstructured data.Data Lakes are very much based on the concept of Apache Hadoop. This is an open-source framework that is designed to store and process large amounts of data on commercially available hardware. For this purpose, the data is processed in a distributed manner on different clusters. Two important components in the Apache Hadoop environment are the Hadoop Distributed File System (HDFS) and Hadoop Yet Another Resource Negotiator (YARN).

HDFS is the Hadoop distributed file system. It is designed to process large amounts of data on standard servers. All files are split into data blocks when stored, and the blocks are stored in redundant form on three different servers by default. This provides high fault tolerance and availability of data, as copies of the data can be accessed in the event of server failures. YARN represents the resource management layer of Hadoop. The available resources of the available hardware are distributed by YARN according to predefined rules. The aim is to ensure that no application uses up the available resources and that other processes cannot use the resources.

Data Lakes were developed due to the changing requirements for data management systems. Previously, it was common to manage data in relational databases in the form of data warehouses. A major difference is the structure in which the data must be available for processing in the data management system. Within a data warehouse, it is necessary to develop complex data models with the data from different sources and to bring these into a homogeneous structure. Subsequently, these can be loaded into the data warehouse and used for analyses or reports. The focus is on the "Schema on Write" process. This means that the structure of the data is precisely defined before it is saved.

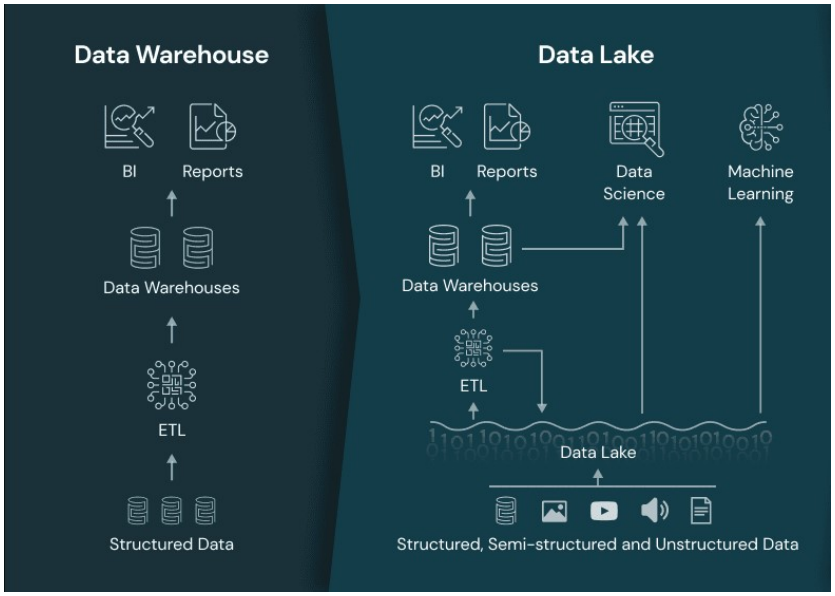

Figure 4.1: Distinction between data warehouse and data lake source:[DWHDL20]

On the other hand, different types of data can be stored on a data lake. The data structure is only defined at the time of use. This process is also referred to as "Schema on Read". With data lakes, a much wider range of data processing methods is also supported, which offers data experts plenty of room to gain new insights. To illustrate this, Figure 4.1 shows a comparison of the structures and possible applications of data warehouses and data lakes.

On data lakes, it is not necessary to apply a concept of standardization or structuring when storing data. This has the disadvantage that data silos are often created in which data is stored in an unorganized manner. In addition, without a conceptualization, legal regulations, such as data protection regulations, can often not be guaranteed. For this reason, the metadata management required in data warehouses becomes more important in data lakes. To avoid such undesirable developments, it is necessary to logically divide data lakes into areas and to record unstructured data with metadata. Thus, data lakes are again increasingly aligned with data warehouses in their conceptual design.

The project described in this thesis is carried out on AWKG's Data Lake. This system solution is provided by IBM and basically consists of two components, the Watson Machine Learning Accelerator (WMLA) and the General Parallel File System (GPFS). Two servers are available on the AWKG data lake for data storage on the GPFS. Similar to the previously described HDFS, the data is stored in distributed manner on several computers. Based on the data from the GPFS, analysis can be performed on the WMLA. WMLA provides an infrastructure through which deep learning and machine learning can be easily accessed and processed on the data lake. This makes it possible to harness the benefits of artificial intelligence in a business context.

As can be seen in Figure 4.2, three servers are dedicated to this area on AWKG's Data Lake. On these, analyses and calculations can be performed. Via WMLA, there is a connection to Jupyter Notebook, where the data analyses are performed in PySpark as well as in Python. In addition to

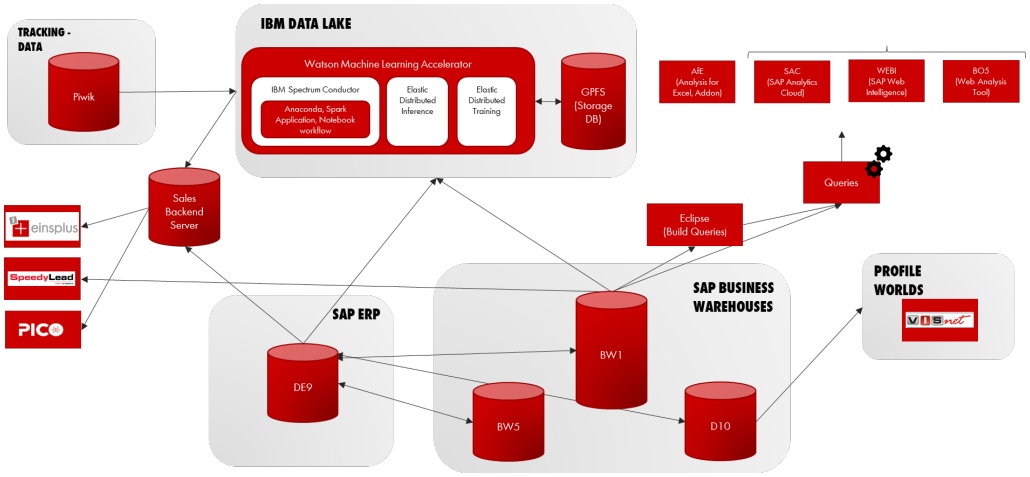

Figure 4.2: System landscape of AWKG (source: Würth internal document)

AWKG's data lake infrastructure, Figure 4.2 shows the source systems' data connections to the data lake. The individual data sources and their connections to the Data Lake are explained in more detail in the following chapter.

4.1.2 Data Models on the Data Lake

In Figure 4.3, we can see all the available data models in wuerth infrastructure. The most frequently used data models on the data lake are invoice and CASA data. For each billing item, there is a row within the data set with all relevant information about that item. The CASA data forms the customer master data record. This is where all the required customer details are recorded.

All other data models provide additional information on more specific topics. Piwik (E-shop data), an open-source alternative to Google Analytics, is used to collect data on user behavior in the Würth online store and app. There is also data on activities on the data lake, which summarizes all interactions between Würth. On the data lake, the data is stored in ORC files. This is a column-oriented data storage format, so that analyses can be performed more quickly and easily. Data from source systems that do not support this format is offloaded as a CSV file and then converted to ORC files on the Data Lake. This process is performed on billing and CASA data. For example, as can be seen in Figure 4.2, this data originates from SAP systems of the company. Since the data is not stored there in column-oriented form, it is transformed into the appropriate format via a script and reformatted. Other data, such as Piwik, can in turn be transferred directly from the source system to the data lake if it is already available in the required format.

4.1.3 Datasets

From the Figure 4.3, CASA is the customer dataset which is used in the project which contains the customer details and will be updated monthly, which is like not frequently updated data and then customer master dataset gets updated daily if there are any changes from the source. Invoice dataset will be having data aggregated for each customer with the sales receipt, ordered date and the

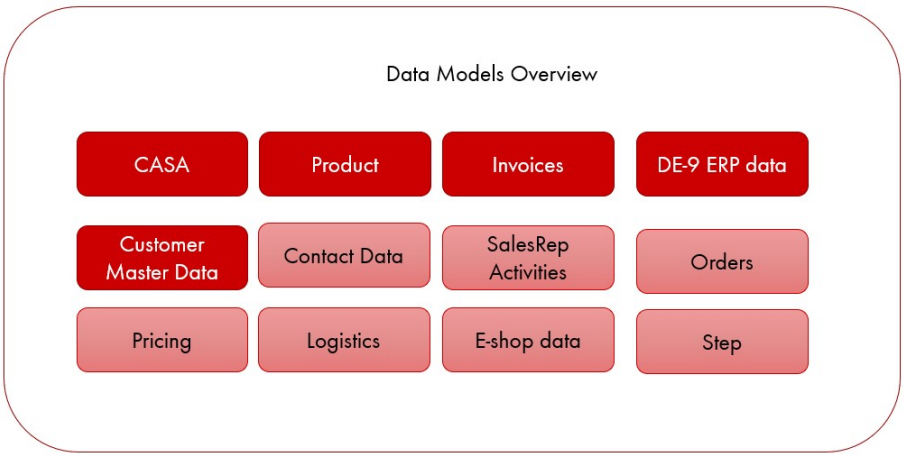

Figure 4.3: Overview of Data models, dark red represents the data model used in the project (source: own representation)

Dataset	Row Value Count (Millions)	Columns Value Count
CASA	2.4M	70
Customer Master Data	2.4M	123
STEP product	2.5M	81
DE-9 ERP Data	2.4M	22
Invoices	360M	121

Table 4.1: Data Description of the data used

shipped date, which sums up the transaction. Product STEP data is versioned data set available to the Germany (DE entity), which contains active and inactive product data. So, for the project we require the active customers and their details also to filter out active products and then join these two datasets to the invoice dataset. So, that the final data set contains invoices of active customers and products, filtering out the inactive records.

The final dataset gets formed based on the filters that are being applied to the CASA dataset for active customers, customers with DEBIT status (customers spending money), and customers who are not marked as deleted, filtering only for the regions from 11 to 18, which are the regions of Germany split according to Wuerth sales strategy. For the product dataset, the active products must be filtered along with the e-shop availability filter. This makes the filtered dataset of the whole chosen datasets, as shown in table 4.1. So, the final data set comprises invoices with active products from the product dataset and active customers from DE-9 ERP data and CASA, and CASA master data. So, the final invoice dataset will have sales values according to customer and their eight divisions, namely, Auto, Construction Site Project, Construction, Cargo, Metal, Wood, House Engineering, and, Engineering Workshop (Betriebswerkstatt), and their regions (11 to 18) and means of sales channel, whether it is through Sales Rep, Telesales, e-shop,etc. We can use this dataset to form a customer-product interaction matrix, which comprises users and their interactions with products. The final dataset prepared is of 104 million (104288146) rows and 25 columns, this will be further modified according to the model.

4.2 Frequent Itemset Mining

Frequent Itemset Mining is a data mining algorithm used to discover frequently occurring itemsets, which are the sets of items that are purchased together. For example, k-itemsets refer to itemsets that contain exactly k items, where k represents the number of items within the itemsets. If we consider k = 3, the number of items in such itemsets would be 3. So, finding the frequent items is dependent on the threshold value set on the parameters, namely, support (Eq. 4.1), confidence (Eq. 4.2), and lift (Eq. 4.3). All these parameters are like hyperparameters to the frequent itemset algorithm, which decides the quality of the final itemsets derived. In the algorithm, we consider items that meet the minimum threshold values of support and confidence. Frequent item set mining is an efficient algorithm for finding the hidden patterns that are in the underlying transactions by a shorter span of time and of reduced memory consumption.

Support(Eq. 4.1) is a measure used to determine the popularity of a product or item within a transactional database. It measures the interestingness of an item by calculating the proportion of transactions that contain the particular item, relative to the total number of transactions in the database.

Confidence(Eq. 4.2) is a metric that measures the strength of an association rule between two sets of items, typically denoted as “A“ and “B“. It represents the probability of a person of buying item “B“ if he bought item “A“. It is calculated as the ratio of the number of transactions that contain all the items in both “A“ and “B“ to the number of transactions that contain all the items in “A“.

The confidence to expected confidence ratio, often known as the **lift** or **lift ratio** (Eq. 4.3). An association rule’s strength and importance are evaluated using the lift metric. It is calculated by dividing the support of itemset B by the confidence of the rule A -> B. Lift measures how much more likely it is for B to occur when A is present as compared to when A is not present. The associations between the elements are greater when the lift values are higher and lower when the lift values are lower.

Frequent pattern mining is used to derive the rules that define the mined relationship from the transaction data. The rule that are derived is called an associative rule. The associative rule is used in many applications, like supermarket basket analysis, finding related words in the document, irregularities in data, and plagiarism detection.

Agrawal et al. [AIS93] defined Association Rule Mining as, “Let I = $\{i_1, i_2, ..., i_n\}$ be a set of ‘n’ binary attributes called items. Let D = $\{t_1, t_2, ..., t_m\}$ be set of transaction called database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A rule is defined as an implication of form X $\Rightarrow$ Y where X, Y $\subseteq$ I and X $\cap$ Y = ϕ. The set of items X and Y are called the antecedent and consequent of the rule, respectively."

$$Support(A) = \frac{\text{Number of transaction in which item A appears}}{\text{Total Number of transactions}} \tag{4.1}$$

$$Confidence\,(A \rightarrow B) = \frac{support\,(A \cup B)}{support\,(A)} \tag{4.2}$$

$$Lift\,(A \rightarrow B) = \left(\frac{\frac{(A+B)}{A}}{\frac{B}{Total}} \right) \tag{4.3}$$

The working of association rule mining is of two steps:

- Find all the frequent itemsets from transactional database and generate candidate set for finding the association rules.
- Generate association rules from the above generated candidate set.

Apriori algorithm is the improved version of frequent itemset mining, which has additional steps such as “join“ and “prune“ for the purpose of reducing the search space in finding the frequent item sets. The apriori algorithm determines the probability of whether the item is frequent or not based on the following conditions:

Condition 1: If the probability of an item is less than the minimum support threshold, then that item isn't frequent.

Condition 2: If the itemset is determined not to be frequent, consequently, any subset of items within that itemset will also be below the minimum support threshold and, therefore, can be ignored. This property is called the antimonotone property.

Apriori algorithm is determined by the following steps

(i). Join step creates (K+1) itemsets from K-itemsets by combining each element with itself.

(ii). Pruning step In this stage, the count of each item in the database is examined. If a candidate item does not meet the minimum support threshold, it is eliminated. Pruning is done to make the candidate item sets smaller. By utilizing the antimonotone property, the Apriori algorithm reduces the search space by avoiding the generation and evaluation of itemsets that are unlikely to be frequent. This pruning steps helps improve the algorithm's efficiency by reducing computational overhead.

(iii). Apriori step is the data mining technique that follows the joining and pruning steps iteratively until the most frequent set of items is found. The steps are clearly depicted in the flow chart as in Figure 4.4. The first minimum support value is initialized, and based on the support value, the frequent itemsets are determined. With frequent itemsets, generate candidate itemsets of length k+1 by combining itemsets that share a common prefix of length k-1. Then the pruning eliminates the infrequent subsets, now with these itemsets, check for the support value and keep only itemsets that meet the threshold value. Repeat these steps with an iterative approach until no more frequent itemsets are generated.

The resulting frequent item sets represent the most frequent sets of items in the dataset, satisfying the minimum support threshold.

With this results we can achieve the rules for product bundling with frequently purchased items together.

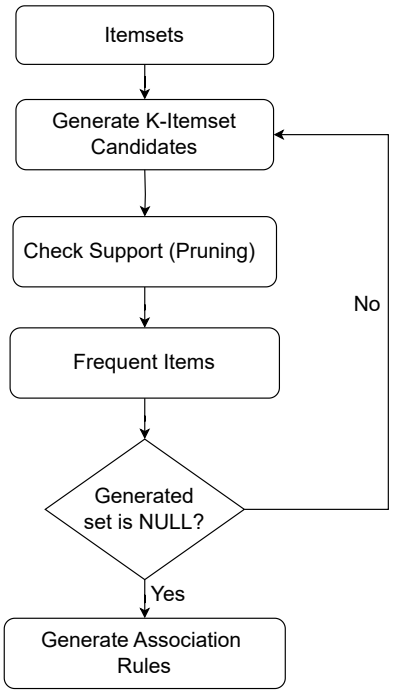

Figure 4.4: Association Rule Mining Flow diagram [SC12]

4.3 Alternate Least Square

Alternate Least Square (ALS), is a machine learning method based on the matrix factorization algorithm. The basic idea of the matrix factorization algorithm is to learn low-dimensional representations of users and items by approximating the original user-item matrix using a lower-rank matrix product. Each row of the user factor matrix represents the user's preference for the latent factors, and each column of the item factor matrix represents the relevance of an item to the latent factors. The multiplication of these matrices approximates the original matrix and allows for predicting missing entries or generating recommendations. This matrix factorization is able to capture the unobserved or hidden patterns within the user-item matrix in lower dimensional space with the help of latent factors. The problem with matrix factorization is that it cannot handle the sparse data and high volumes of datasets, in this case the algorithm performs poorly and fails to capture the pattern. For the improvement of performance in handling the sparse data of the user-item matrix and handling the data in parallel for better resource utilization, the alternate least square [TT12] method is proposed.

Alternate least squares method is different from matrix factorization, it works by factorizing a matrix iteratively to solve least squares problems. The user-item matrix is decomposed into two lower-rank matrices, one with the user preferences and another with the item attributes. The ALS algorithm works by initializing the random or initial values for the user and item matrices. Next, the following steps are repeated iteratively, the user matrix U is kept fixed (Eq: 4.4) and the item matrix is solved for the minimization of least square problems (Eq: 4.5) and with the item matrix V fixed (Eq: 4.6), the user matrix is solved for the minimization of least square problems (Eq: 4.7). In each iteration the loss function gets reduced or stays the same but never increases. This helps out in convergence to local minima and, depending on the value of the user or item initial values,

guarantees the reduction of loss functions (Eq: 4.5 & Eq: 4.7) in the iterative approach and seeks to find the best-fitting factorization that represents the original matrix. The resulting lower-rank matrices can be used to generate recommendations for the user. In the Equations: 4.4,4.5,4.6, and 4.7 R is the original user-item matrix, U is the user matrix, V is the item matrix, $\| \ . \ \|^2$ represents the squared Frobenius norm, and $\lambda_u * \|U\|^2$ is a regularization term that prevents overfitting.

By using the ALS method, we can efficiently capture the hidden patterns and provide recommendations to the customer, considering the memory constraint and the cold start problem.

$$U = RV^T(V^TV + \lambda_u I)^{-1} \tag{4.4}$$

$$Loss(U) = \underset{U,V}{argmin} \parallel R - UV^T \parallel^2 + \lambda_u \parallel U \parallel^2 \tag{4.5}$$

$$I = RU^T(U^TU + \lambda_v I)^{-1} \tag{4.6}$$

$$Loss(I) = \underset{U,V}{argmin} \parallel R - UV^T \parallel^2 + \lambda_v \parallel V \parallel^2 \tag{4.7}$$

4.4 Item-Item Collaborative Filtering

Item-Item Collaborative Filtering recommends the users with the items based on their past preferences. If there are two items and we need to compute the likeness of the users to these two items, we need to calculate the similarity between these two items. Similarity is calculated by squaring the ratings, then multiplying the ratings of those two items by the whole user base, then summing them and taking the square root to get the final results. We will end up having two values for two items. With these, divide the results accordingly if the first item similarity should be calculated with the second item, then the first item score must be divided by the second score, and then it will lead to a single score, providing the similarity of item 1 to item 2. This process will be repeated until the similarity scores are calculated for all items. Item-Item Collaborative Filtering is mathematically written as in equation 4.8, where in similarity(i,j) i is the main item for which we need to find the other similar items, j is the item that gets compared with the main item "i" to find if they are similar, $r_{(u,i)}$ user "u" rating for item "i", $r_{(u,j)}$ user "u" rating for item "j", $\sum_u^U$ represents the sum of the multiplication of ratings of "i" and "j" given by all users U, $\sqrt{\sum_u^U r_{(u,i)}{}^2}$, $\sqrt{\sum_u^U r_{(u,j)}{}^2}$ represents the multiplication of square roots of the sum of squared ratings of item "i" and "j" by user "u". This will create a table of similarity scores for each item. From this which items should be recommended to users from the list of similar items should be decided.

For this purpose, we use users' past history of rated items to generate the recommendation. This is calculated mathematically by the equation 4.9, where $\bar{r}_i$ represents the sum with user i's average rating, $\sum_j^I similarity(i,j)$ provides the sum of item i and j similarity scores, $r_{(u,j)} - \bar{r}_j$ denotes subtracting with average rating to normalize the rating scale, $\sum_I^j$ denotes the sum of the multiplication of item i and j's similarity and difference of rating by users "u" and "j" and its average rating. From this, the top scores with items per user will be generated, and we can recommend products based on this list to the user.

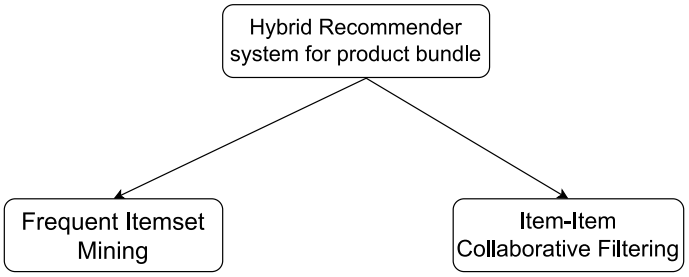

Figure 4.5: Hybrid recommendation system for product bundle with frequent itemset mining and item-item collaborative filtering.

$$similarity(i,j) = \frac{\sum_u^U r_{(u,i)} r_{(u,j)}}{\sqrt{\sum_u^U r_{(u,i)}{}^2}\sqrt{\sum_u^U r_{(u,j)}{}^2}} \tag{4.8}$$

$$score(u,i) = \frac{\sum_I^j similarity(i,j)(r_{(u,j)} - \bar{r}_j)}{\sum_j^I similarity(i,j)} + \bar{r}_i \tag{4.9}$$

Since recommender system will have issues like cold start problem and further issues which makes to skip certain users to not get recommended with products only with one method. For this we need to include another method to solve with the edge cases that are left behind. So, we are using hybrid recommender system for the product bundling. With the advantage of frequent itemset mining and item-item collaborative filtering we combine them to retrieve the product bundles as hybrid recommender product bundling system as in figure 4.5.

4.5 Neural Collaborative Filtering

Matrix factorizations' performance is hindered by the inner product of the user and item matrix, as it fails to capture the features of the underlying pattern. To overcome this drawback, the Neural Collaborative Filtering (NCF) method is introduced. It is a deep learning method that aids the user-item inner product of the matrix factorization method with the neural architecture by introducing the bias terms into the inner product computation. With a multi-layer perceptron (MLP) architecture, NCF enables the modeling of user-item feature interactions and the learning of continuous functions. The MLP's multiple layers facilitate the capture of high levels of non-linearities, enhancing its ability to learn complex user-item interaction functions. This makes NCF well-suited for effectively modeling and predicting user-item interactions. Generally, matrix factorization works based on the scalar product of user-item latent vectors mentioned mathematically with the help of the following equations: (Eq. 4.10 & Eq.4.11),

$$\widehat{y_{ui}} = f(u,i|p_u,q_i) = p_u^T q_i \tag{4.10}$$

$$\widehat{y_{ui}} = \sum_{k=1}^{K} p_{uk} q_{ik} \tag{4.11}$$

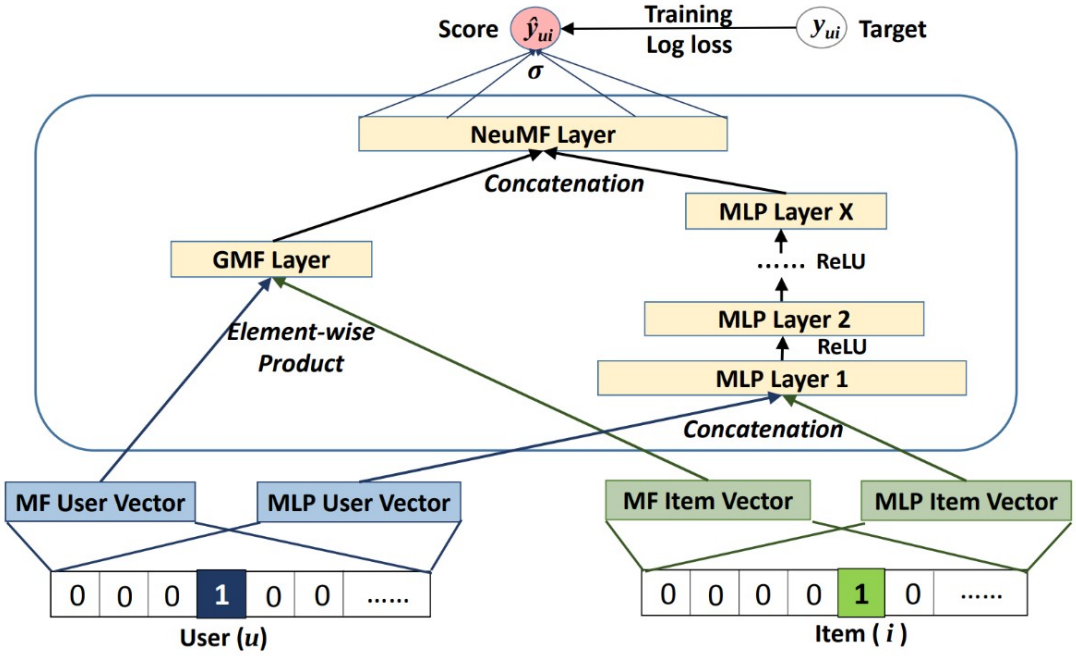

Figure 4.6: Neural Collaborative Filtering Framework [HLZ+17]

In both the equations 4.10 & 4.11, the y_{ui} represents the prediction score for the interaction between user u and item i, p_u represents the latent vector for user u, q_i represents the latent vector for item i, and K represents the dimension of latent space. As matrix factorization with the scalar product of user-item latent vectors makes the user and item latent vectors be represented in the same latent space. This representation is made with the similarity measure between the users, which is calculated with the help of the cosine of the angle between their latent vectors. With this, the matrix factorization is able to place similar users near each other, but the placement of the user near to the another user in the latent space will make the ranking differences and the recommended products not relevant for the users. To solve this issue, we are using the Neural collaborative filtering approach of combining the multi-layer perceptron (MLP) for learning the interaction functions, which are more non-linear, from the user-item data.

From the figure 4.6, the NCF model behaves as an ensemble method by combining the general matrix factorization (GMF) and multilayer perceptron (MLP) outputs to concatenate and finalize the recommendation. General matrix factorization (GMF) and Multilayer perceptron (MLP) have separate user and item embeddings as the initial layer, for the purpose of learning embeddings separately. With this embedding layer, the sparse representation of the user-item vector is converted to a dense vector. This representation becomes the latent user and item vectors. General matrix factorization (GMF) is similar to matrix factorization by doing the element wise scalar product of the user-item vector. Multilayer perceptron (MLP) takes the concatenation of the user-item latent vectors as input and passes those values to stacked hidden layers. NeuMF (Neural Matrix Factorization) layer combines the output from the general matrix factorization (GMF) and the

multilayer perceptron (MLP) to form the final prediction score. Mathematically, neural collaborative filtering modifies the Matrix factorization equations 4.10 & 4.11 by the equation 4.12, where P represents the latent factor matrix for users (size = M × K), M represents the number of users, K represents the latent factors, Q represents the latent factor matrix for items (size = N × K), N represents the number of items, and Θ_f represents the MLP model parameters.

$$\widehat{y_{ui}} = f(P^T v_u^U, Q^T v_i^I | P, Q, \Theta_f) \tag{4.12}$$

$$f(P^T v_u^U, Q^T v_i^I) = \Phi_{out}(\Phi_X(...\Phi_2(\Phi_1)(P^T v_u^U, Q^T v_i^I))...)) \tag{4.13}$$

Multilayer perceptron function is represented in equation 4.13,where Φ_{out} is the mapping function for the output layer and Φ_X is the mapping function for the x^{th} neural collaborative filtering layers. This acts as a scoring function for neural collaborative filtering (NCF). Neural collaborative filtering (NCF) is an example of multimodal deep learning that has both GMF and MLP models in it. Initially, data from user and item inputs was combined through the concatenation layer. But the concatenation of user and item vectors doesn't learn the whole pattern for the collaborative filtering effect. So, the Multilayer perceptron (MLP) has hidden layers added to concatenated user-item interactions, which helps to learn user-item interactions non-linearly along with linear interactions being captured by GMF.

Multilayer perceptron changes the generic recommendation algorithm equation 4.14 modeling part $f(u, i|\Theta)$ into the equation 4.15, where W represents weight matrix, b represents bias vector, a(x) represents activation function for the x^{th} layers' perceptron, p and q represent latent vectors for the user and item, respectively, and h represents the edge weights of the output layer.

$$\widehat{y_{ui}} = f(u, i|\Theta) \tag{4.14}$$

$$\begin{aligned} z_1 &= \Phi_1(p_u, q_i) = \begin{bmatrix} p_u \\ q_i \end{bmatrix}, \\ \Phi_2(z_1) &= a_2(W_2^T z_1 + b_2)), \\ &..., \\ \Phi_L(z_{L-1}) &= a_L(W_L^T z_{L-1} + b_L)), \\ \widehat{y_{ui}} &= \sigma(h^T \phi_L(z_{L-1})) \end{aligned} \tag{4.15}$$

Because of the multiple hidden layers in the multilayer perceptron (MLP), the model sufficiently learns the user-item interactions in comparison with the matrix factorization of the fixed element-wise product of the latent vectors. NeuMF (Neural Matrix Factorization) layer concatenates the output from general matrix factorization (GMF) by scalar multiplication and MLP with multiple neural layers with non linear interactions and then feeds into the last dense layer to form prediction results, represented with equations 4.16, where Φ^{GMF} represents the GMF output, Φ^{MLP} represents the MLP output, and $\widehat{y_{ui}}$ represents the final concatenation layer with sigmoid activation function.

$$\begin{aligned} \Phi^{GMF} &= p_u^G \odot q_i^G, \\ \Phi^{MLP} &= a_L(W_L^T(a_{L-1}(...a_2(W_2^T \begin{bmatrix} p_u \\ q_i \end{bmatrix} + b_2)...)) + b_L), \\ \widehat{y_{ui}} &= \sigma(h^T \begin{bmatrix} \Phi^{GMF} \\ \Phi^{MLP} \end{bmatrix}), \end{aligned} \tag{4.16}$$

With the advancements in their way of solving for providing the recommendations that are much related to the user with the consideration of sparsity of the user-item interaction matrix, ability to solve the task in parallel, and combining the linear matrix factorization with non-linear multi-layer perceptron to learn the user-item interaction in detail, NCF and ALS methods are considered to be perfect suggestions for the recommendation with the ability of personalization and resource efficiency.

4.6 Dimensionality Reduction Techniques

For an understanding of how the neural collaborative filtering predicts the results with the user features being fed into it, we are using dimensionality reduction methods as in Figure 4.7. To understand the learning of the model in the final layer of multi layer perceptron (MLP), general matrix factorization (GMF) and Neural Matrix Factorisation (NMF) we take the output from the final layer and process it to find the relationship between them.

In this thesis, we will discuss dimensionality reduction techniques like t-SNE (t-Distributed Stochastic Neighbor Embedding), Principal Component Analysis (PCA), Uniform Manifold Approximation and Projection (UMAP) and multidimensional scaling (MDS) methods. Through a non-linear reduction of the dimensions of multi-dimensional data to two or three dimensions, the t-SNE algorithm makes multi-dimensional data more observable by humans. Its primary goal is to accurately maintain local similarities or small distances between data points while also preserving the data's overall structure. In order to do this, t-SNE computes similarity measures in both the original high-dimensional space and the reduced low-dimensional space, and then optimizes these measures based on a cost function.

Three steps are required to apply dimensionality reduction methods to a dataset, such as t-SNE. The first step is to compare data points in high-dimensional space. For instance, a Gaussian distribution is built around each point in a 2-D space with dispersed data points, and the density of points under each Gaussian distribution is calculated. As a result, all of the data points have a set of probabilities that are proportional to how similar they are. When two points, denoted by the symbols a_1 and a_2, have equal areas under the Gaussian distribution, it means that their local structures in the higher-dimensional space are similar.

The following are the parameters that are used for the t-SNE optimization: Learning rate is used to control the step size of the gradient updates. Perplexity can be used to change the variance and the number of nearest neighbors of the Gaussian distribution used in the initial step. Comparable to the first step, the second step computes probabilities in the lower-dimensional space using a Cauchy distribution as opposed to a Gaussian distribution. Early exaggeration factor is used to

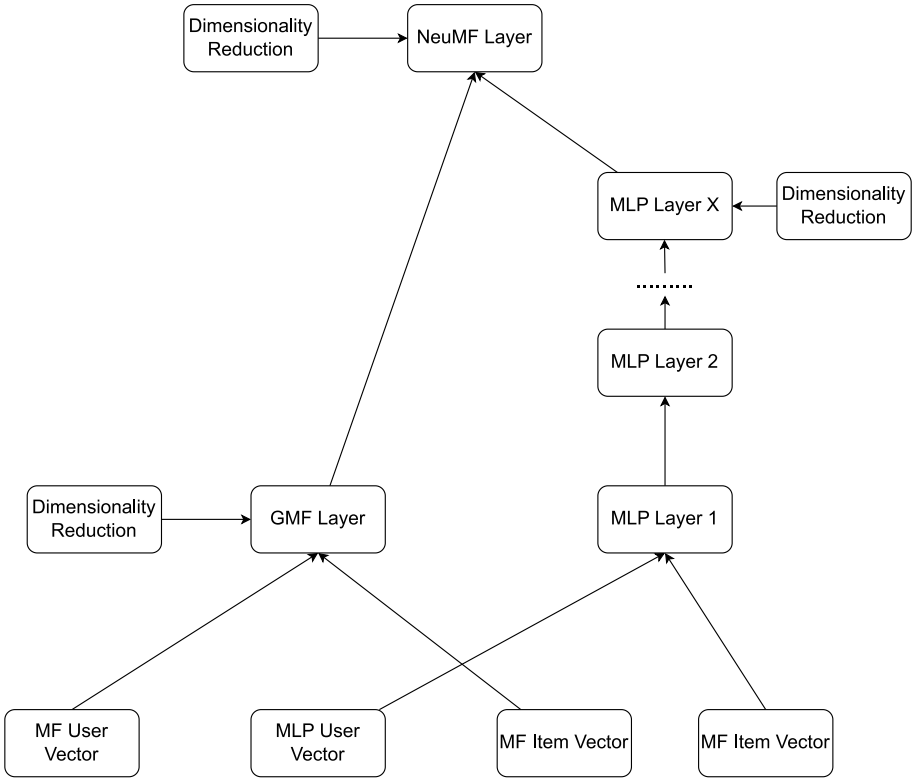

Figure 4.7: Applicability of dimensionality reduction in Neural Collaborative Filtering (NCF) approach

increase the attractive forces between points and allows points to be freely moved along to find their nearest neighbors more easily. This helps make the separation between the clusters more obvious. The final goal of the t-SNE algorithm is to minimize the difference between the probabilities of the low-dimensional space and the high-dimensional space. It is common practice to gauge the difference in probabilities between the two spaces using the Kullback-Leibler (KL) divergence. With this, we can compare the lower and higher-dimensional spaces.

Principal Component Analysis (PCA), is the method for reducing the dimensionality of large datasets by representing a large set of variables into a smaller one that contains the information from the large dataset used while preserving as much information as possible. PCA follows the following steps: first, the dataset is standardized to make all the variables on the same scale in order to avoid biased results because the values of variables with large values dominate the values of variables with a small range of values. After this, the covariance matrix is computed. It's based on how the variables of the data set vary from the mean with respect to each value in order to obtain the relationship between the features used. The covariance matrix is an a $\times$ a symmetric matrix, where a is the number of dimensions that have entries for all possible pairs of the initial variables. If the value represented in the matrix is positive, then the two variables are correlated, and if it is negative, they are inversely correlated. After this, the eigenvectors and eigenvalues of the covariance matrix are computed in order to determine the principal components of the data. Principal components are the variables that represent the linear combinations or mixtures of the provided features. With the number of features to be represented, the PCA calculates the optimal values based on the eigenvalue

and vector and then ranks them accordingly in descending order with high variance values. The principal components are represented by the order of ranking, which has significant values. The uncorrelated features are represented by the principal components in the result by removing the most correlated features to preserve the data that is contributing to the representation.

Uniform Manifold Approximation and Projection (UMAP) is used for non-linear dimension reduction, similar to t-SNE. But UMAP builds a high-dimensional graph representation of the provided data and then optimizes a low-dimensional graph to be as structurally similar as possible in order to preserve the data. The high-dimensional graph is based on the weighted graph of edge weights to represent the likelihood that two points are connected. The likelihood is calculated based on the radius of the neighbors near the point that's being represented. So, the hyperparameters number of neighbors and minimum distance play pivotal roles in high-dimensional graph representation. Finally, by stipulating that each point must be connected to at least its closest neighbor, UMAP ensures that local structure is preserved in balance with global structure.

Multidimensional scaling (MDS) is another dimensionality reduction system that represents the data as dissimilarities between the data points provided. It's used to represent these high-dimensional data as distances between points in a low-dimensional space, so that the distances are as close as possible to the observed data points. MDS system is applied to relational data, that is, data with correlations, distances, closeness, parallels, multiple standing scales, and preference matrices. The problem with this approach is the demand for the shape of the matrix to be square.

With these dimensionality reduction methods, we can understand the learning pattern or relationship between higher-dimensional data and lower dimensional data. Since visualizing the higher-dimensional data to derive insights without dimensionality reduction is a tedious process without a clear understanding of the representation. With the advent of low-dimensionality reduction techniques, we can visualize the outputs from each final layer of the neural collaborative filtering approach and find the similarities.

In these dimensionality reduction methods, the most predominant method is t-SNE because of its ability to preserve the data without losing the relationship in 2D representation. It's made possible by the ability to hold the local variance by retaining the variance of neighboring points. PCA is fast, simple, and retains the overall or global variance of the dataset. But it has the drawback of not holding the non-linear variance. UMAP is preferred because it holds the local structure of the dataset. MDS is chosen because of its ability to represent data points by reducing the distances between input and output spaces.

4.7 Model interpretability & K-means Clustering

4.7.1 Model interpretability

Model interpretability is an important research topic in the machine learning world. There are multiple methods that explain the results prediction of the model by providing the feature importance that contributes more towards the output deciding factor. Two commonly used model interpretability methods are LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations).

LIME is a model-agnostic method that provides local explanations for individual predictions. LIME perturbs the features of the instance and observes the resulting changes in the predicted outcome. By analyzing the local model's coefficients or feature importance, LIME identifies the features that have the most influence on the prediction. SHAP is a unified framework rooted in cooperative game theory that assigns importance values to features based on their contribution to the prediction of a specific instance. SHAP values quantify the impact of each feature by considering all possible combinations of features and measuring the change in prediction when a feature is included or excluded.

With these methods, we can apply the deep explainer method from SHAP in NCF to get the feature for which the particular data gets weighted on each layer. It simply does the back propagation to find the local minima and find the best feature on trace back. This provides features that contribute more to the output and are helpful to check the model's interpretability.

4.7.2 K-means Clustering

As mentioned in Chapter 2 about unsupervised learning, the purpose of this algorithm is to group or cluster the data points of similar patterns. A cluster is a collection of data points aggregated based on similar characteristics. K-means is based on the "centroids" of each data point with K-group. The number of centroids determines the number of clusters to be formed later. A centroid is the imaginary or real point that represents the center of the cluster. Later, it assigns each data point to a cluster based on its closest centroid. The process starts with randomly selected centroids, which are the beginning points for every cluster, and then performs repetitive calculations to optimize the positions of the centroids. After optimizing the centroids to the stabilization point where there is no change in the values, clusters are formed.

After forming the clusters, we could find broad distribution patterns and relationships across data variables as well as dense and sparse areas in object space. Though the method is commonly used, it has problems with the results when there is a slight change in the data, leading to a high variance. But this algorithm performs well for high-dimensional data. By using K-means clustering, we could form the clusters of the low-dimensional representation of each data point. The outputs of each layer are represented by user and item interactions; by reducing the item interactions to a low dimension, we could form a non-linear representation of the user-item relationship. On this low-dimensional representation, we can apply K-means clustering and then cluster each user into that group and confine them to a group of users where the users are getting suggested items based on that user group.

To find the optimal number of clusters (K), we can use two methods, namely, Elbow method and Silhouette Score. Elbow method is used to find the optimal point where the dataset gets linear and does not have abnormal changes. This works by calculating the sum of the squared distance between each data point and its corresponding cluster centroid. An elbow plot is plotted with the sum of squared distances against the number of clusters, and the point where the values get diminished, reducing the sum of squared distances, forms an elbow shape. This depicts that as the number of clusters increases,the sum of squared distances decreases. With this intuition, the elbow point can explain the data effectively by minimizing the sum of squared distances. We have used the Sklearn library's inbuilt methods, KMeans, to calculate the elbow plot.

Silhouette score works based on the distance between neighboring clusters and also on the points inside the cluster itself. It is used to find how close a point in one cluster is to another cluster. The silhouette score range is from -1 to 1. Higher the silhouette score, the better the clusters are at representing the well-defined clusters; lower the silhouette score, the distance is in the boundary and the points are overlapping each other by not representing the points properly. We have used the sklearn library's inbuilt methods silhouette_samples and silhouette_score for the silhouette score calculation.

5 Evaluation

The goal of this thesis is to build a product bundling recommendation system from scratch by understanding business and data. Then, to build a visualization toolbox to support the results that are derived from the machine learning methods applied for getting the results and supporting business users. In this chapter, we discuss the development environment, software design, benchmark results, comparisons, and details about various visualizations in the toolbox. Section 5.1 deals with the experimental setup, hardware, and software design used for developing this tool. Section 5.2 provides the evaluation metrics that are used for evaluating the performance of Alternate Least Square (ALS) and Neural Collaborative Filtering (NCF). Section 5.3 gives an overview of the comparison of Alternate Least Square (ALS) and Neural Collaborative Filtering (NCF) results and bench-marking for better recommender model selection. Section 5.4 gives an overview of the architecture of the Neural Collaborative Filtering (NCF). Section 5.5 deals with the features in the visualization tool.

5.1 Experimental Setup and Software Design

5.1.1 Hardware

Data lake data, as mentioned in Section 4.1, are stored and maintained in IBM Spectrum Scale GPFS (General Parallel File System), which is composed of a large number of small servers with hard disks, IC 922 hard disk is responsible for organization of the metadata, loading processes, management of servers, and management of requests, while the LC and AC 922 hard disks are responsible for Master Node and worker nodes in Spark Cluster, respectively. The visualization toolbox is built on a PC with an Intel(R) Core(TM) i5-8365U processor, 8 GB of RAM, and the Windows 10 64-bit Enterprise edition operating system.

5.1.2 Software

NCF model is trained with open source framework Tensorflow 2.4.1, used for building scalable models of machine learning and artificial intelligence using Python 3.7.10. ALS model and frequent itemset mining are trained with pyspark 3.4.0 machine learning library of recommendation algorithm and parallel FP-growth algorithm respectively. Used inbuilt scikit library for applying dimensionality reduction methods like t-SNE, PCA, MDS, and UMAP. Flask, networkx, and D3.js are used for visualization tool front end and back end. Pycharm IDE was used for development purposes.

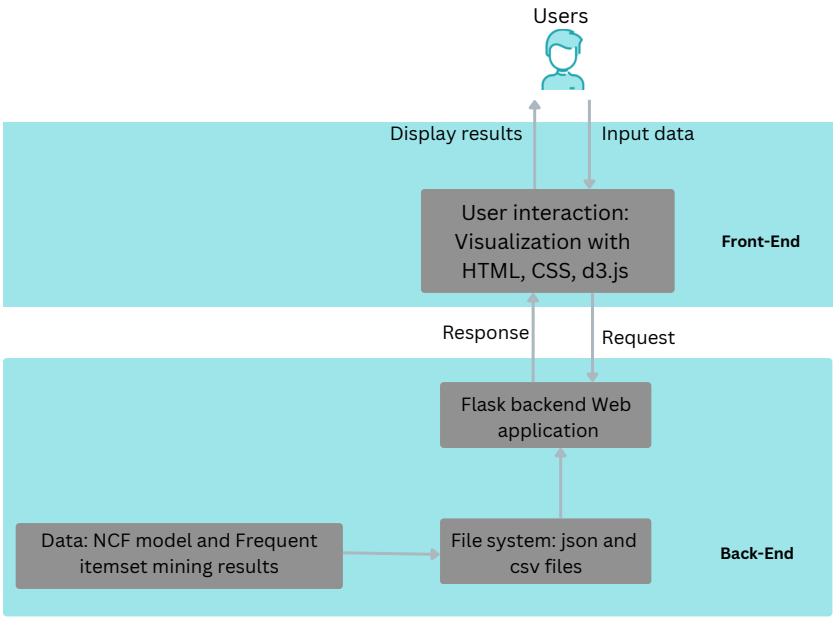

Figure 5.1: Software design of the visualization toolbox

5.1.3 Software Design

The visualization toolbox is designed to have a user interface and to display according to the choices that the user makes. This is handled with python library Flask in the backend to serve the users' requests. For responding to user requests through front-end visualizations, HTML (Hyper Text Markup Language), javascript library D3.js (Data Driven Document), and Cascading Style Sheet (CSS) are used.

Used Pyspark and Python libraries to train frequent itemset mining and neural collaborative filtering, respectively. Those outputs that are required to display the necessary information in the visualization toolbox are stored as JSON and CSV files. As in the figure 5.1, the user requests the necessary information using the front end as input, and that input is handled by the flask web application through the HTTP POST request, and the necessary information is displayed with the PULL response. So, when the user interacts with the tool, each request will be handled by flask and then respective visualizations will be pulled in on the front end with the corresponding HTML/CSS files with interactions. There are multiple HTML files to handle each request from the users.

5.2 Evaluation Metric

5.2.1 Root Mean Square Error (RMSE)

One of the most commonly used methods to assess the quality of the predicted output in machine learning is the Root Mean Square Error (RMSE) or root mean square deviation. It measures the difference between the predicted and true values using the Euclidean distance. RMSE is computed by calculating the difference between prediction and ground truth for each data point; the mean of

the normalized value for each data point of the difference is computed, and the square root of the mean value is taken. RMSE is expressed in equation 5.1, where N is the number of data points, $y(i)$ is the i^{th} measurement, and $\widehat{y}(i)$ is its corresponding prediction:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \widehat{y}(i)\|^2}{N}}. \tag{5.1}$$

RMSE provides the score in the range of 0 to infinity. If the RMSE values are near zero, the model performs well and fits the dataset; if they are inclined towards higher values, the model's performance should be improved. Since the errors are squared before they are averaged, RMSE results provide high weightage to the large errors, which makes RMSE undesirable in cases of large errors between the prediction and actual true values. Because of the squaring of the error values, the result will always be positive. With the help of the square root, the squaring effects will be compromised. We evaluate the NCF and ALS models with RMSE values to assess their performance.

5.2.2 Hit Ratio

For evaluating the results of the recommender systems, we need to evaluate the recommended outcome that they produced. Generally, the recommender systems' top N products will be suggested to the customers. Based on necessity, the N value will be decided. On the top-N recommended products, how well the recommendation system generated the recommendations for the user is evaluated with the help of the hit ratio metric. It is calculated by considering the test data, which consists of items that have already been rated by the user and are also present in the top-N recommended items list but removed in the training data. Based on the N value, the hit ratio value is calculated. For instance, if N = 5, there will be 5 products recommended to the user, hit ratio metric is calculated by taking how many items that have already been purchased by the user are present in that recommendation list. If hit ratio is high, the model performance is good; if it's low, the model results aren't good enough. We use leave-one-out cross validation here to intentionally remove certain products that are purchased by the customer, and in the results of the hit ratio, it will be high if the removed results are present in the recommendation. Hit ratio is formulated as in equation 5.2, where $\left|U_{hit}^{L}\right|$ is the number of users for whom the correct recommendation is included in the top N recommended products and $|U_{all}|$ is the total number of users in the test dataset.

$$HitRatio = \frac{\left|U_{hit}^{L}\right|}{|U_{all}|} \tag{5.2}$$

5.2.3 Normalized Discounted Cumulative Gain (NDCG)

Normalized Discounted Cumulative Gain (NDCG) measures how relevant the items that are recommended are in the order they appear. It is calculated with the Discounted Cumulative Gain (DCG) and Ideal Discounted Cumulative Gain (IDCG). Discounted Cumulative Gain (DCG), see equation 5.3, is calculated for placing the highly relevant items at the top of the recommended list. The drawback is that it depends on the length of the recommended products in the results,

which affects the ranking score. As for the higher ranks, the DCG score will be better because of the length of the recommendation. This disadvantage can be overcome with the Ideal Discounted Cumulative Gain (IDCG), see equation 5.4, which ranks the items top-down according to their relevance up to position k. Normalized Discounted Cumulative Gain (NDCG), equation 5.5, is finally deduced by normalizing the DCG score by the IDCG score, resulting in a range of 0 to 1, irrespective of the length. As in equations (5.3, 5.4, 5.5), I(k) represents the ideal list of items up to position k, $|I(k)|$ = k, G_i represents the gain for each item in k.

$$DCG(k) = \sum_{i=1}^{k} \frac{G_i}{log_2(i+1)} \tag{5.3}$$

$$IDCG(k) = \sum_{i=1}^{|I(k)|} \frac{G_i}{log_2(i+1)} \tag{5.4}$$

$$NDCG(k) = \frac{DCG(k)}{IDCG(k)} \tag{5.5}$$

5.3 Recommendation Model Benchmark Results

From the final invoice dataset prepared in Section 4.3, which contains 104 million records of active customers and products, we create the user-item interaction matrix by determining the rank of each interaction through manual calculation. This is necessary because we do not have any explicit ratings available for the interactions. We calculate the implicit ratings by comparing each customer's buying frequency of a product to the buying frequency of a product by the whole customer. Because of the high amount of data, we consider the data to be further split into eight customer areas, namely Auto, Construction, Cargo, House Engineering, Metal, Wood, ConstrSite project, and Engineering workshop (Betriebswerkstatt), and filtered to one year from the current date for applying the recommendation algorithm. With this pre-processed data, we can form a user-item interaction matrix with the user, the item, and the interaction for each item. ALS and NCF models are trained after hyperparameter optimization with the mentioned hyperparameters in the tables 5.1 and 5.2. For the results depiction, we use the models that are trained for the construction customer area with the hyperparameters mentioned in the tables 5.1 and 5.2 are evaluated with the three metrics mentioned in the section 5.2. The evaluation metrics RMSE, hit ratio, and NDCG help to benchmark the model's performance by comparing the results produced by the model.

RMSE values of the ALS and NCF methods are defined in the table 5.3. From this, we can infer that the RMSE values near zero perform well in accordance with fitting the model and providing better recommendations. So, the RMSE value of 0.0231 for NCF is better than the RMSE value of 0.2598 for ALS. But generally, both values are considered good according to the general RMSE computation.

The hitratio values of the ALS and NCF methods are defined in the figure 5.2 (1). From the figure, we can deduce that NCF performs better in comparison with ALS. The top k values that contain the product that the customer already purchased but that wasn't provided as input in the training data are

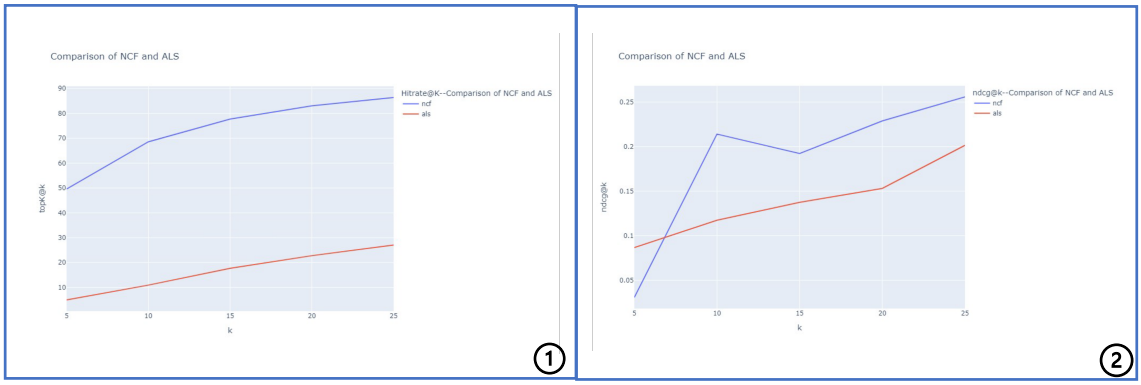

Figure 5.2: Hit ratio and NDCG of NCF and ALS displayed for five different k values **(1):** Denoting the hit ratio calculation on top k products suggested to the customer; **(2):** Denoting the NDCG calculation on top k products suggested to the customer

ALS hyperparameters	**Values**
Rank	70
Max Iteration	50
Regulation Parameter	0.15
Alpha	1
nonnegative	True
implicitPrefs	True
coldStartStrategy	drop

Table 5.1: ALS model hyperparameters which are tuned with grid search method and the current output depends on this parameters

NCF hyperparameters	**Values**
latent dimension	32
MLP layer size	[64, 32, 16, 8]
Regulation Parameter	0.0001
Dropout Rate	0.1
Activation Function	Relu
Learning Rate	0.0001

Table 5.2: NCF model hyperparameters which are tuned with grid search method and the current output depends on this parameters

Method	**RMSE**
ALS	0.2598
NCF	0.0231

Table 5.3: RMSE Values of Alternate Least Square (ALS) and Neural Collaborative Filtering (NCF) methods

present more in the NCF results than the ALS results. For instance, for the top 5 products suggested, the ALSs' and NCFs' hit ratio scores are 5.98 and 49.594, respectively. This clearly depicts the gap between the performance of ALS and NCF, which conveys that the performance of NCF is better.

NDCG values of the ALS and NCF methods are defined in the figure 5.2 (2). From the figure, we can deduce that NCF performs better in comparison with ALS. The order of the top k values that contain the product that the customer already purchased but that wasn't provided as input in the training data is present more in the NCF results than the ALS results. For instance, for the top 5 products suggested, the ALS's and NCF's NDCG scores are 0.086611 and 0.030728, respectively. This shows that the product being suggested to the customer is in the order of the highest ranking for ALS initially. But for the next k instances, the value seems to be increasing for NCF compared to ALS. This depicts that the number of products getting suggested to the customer by ALS didn't have the relevant products; in the case of NCF, it suggests the products that are of the customer's preference.

Overall, the performance of the NCF model is better in comparison with ALS, as per the benchmarked results with the metrics. So, we are considering the NCF model to be further analyzed with the model interpretability.

5.4 Architecture of NCF Model

After benchmarking the results, as per our requirements, NCF performed well in comparison with the alternate least square method. So, we built our visualization tool to make the results from the NCF model post-explainable. He et al. [HLZ+17] proposed neural collaborative filtering to include the neural architecture to supercharge the Collaborative filtering model with non-linear learning of user-item interactions along with the general matrix factorization, which is the inner product of users and items. Figure 5.3 shows the NCF architecture from the input to the final output layer and the dimensions of each layer. After trying out different latent vector parameters, we have fixed 32 as the latent vector dimension. This means the input for the model on each side of GMF and MLP is a representation of the user and item input in a low-level dimensional space of 32 dimensions. We could see this clearly from the figure in the first layer, the input is of 1 dimension, and with the embedding layer in the second layer, the output is 32 dimensions of each user and item input that are being passed as input to the flatten layers. In flattening layers, the three-dimensional values are converted to two-dimensional values. Here, none represents the value of the batch size getting passed to the layers in each epoch. After flattening the inputs from each layer, the user and item inputs are multipled in the GMF (left side of the figure 5.3). As the GMF principle is the inner product of user and item vectors, The user and item vectors are multiplied in the multiply layer. We could see the input from each user and item as 32 values, and the multiplication output as also 2D with batch size and 32 values. On the other side, for NCF, the flattened input of user and item is fetched to the concatenated layer, where the user and item values are concatenated to form 2D of batch-size inputs and 64 values. This will be passed to the consecutive layers of 64, 32, 16, and 8 hidden neurons or units. The values will be transformed in each hidden layer according to the number of neurons processing the inputs, and the output will be fed forward as input to each hidden layer. The output layer is a concatenation of the outputs of the GMF and MLP. Here, the batch size is fixed at 128 after hyperparameter optimization. The output of GMF and MLP is (batch size, 32) and (batch size, 8), which are concatenated to form the output with the shape of (batch

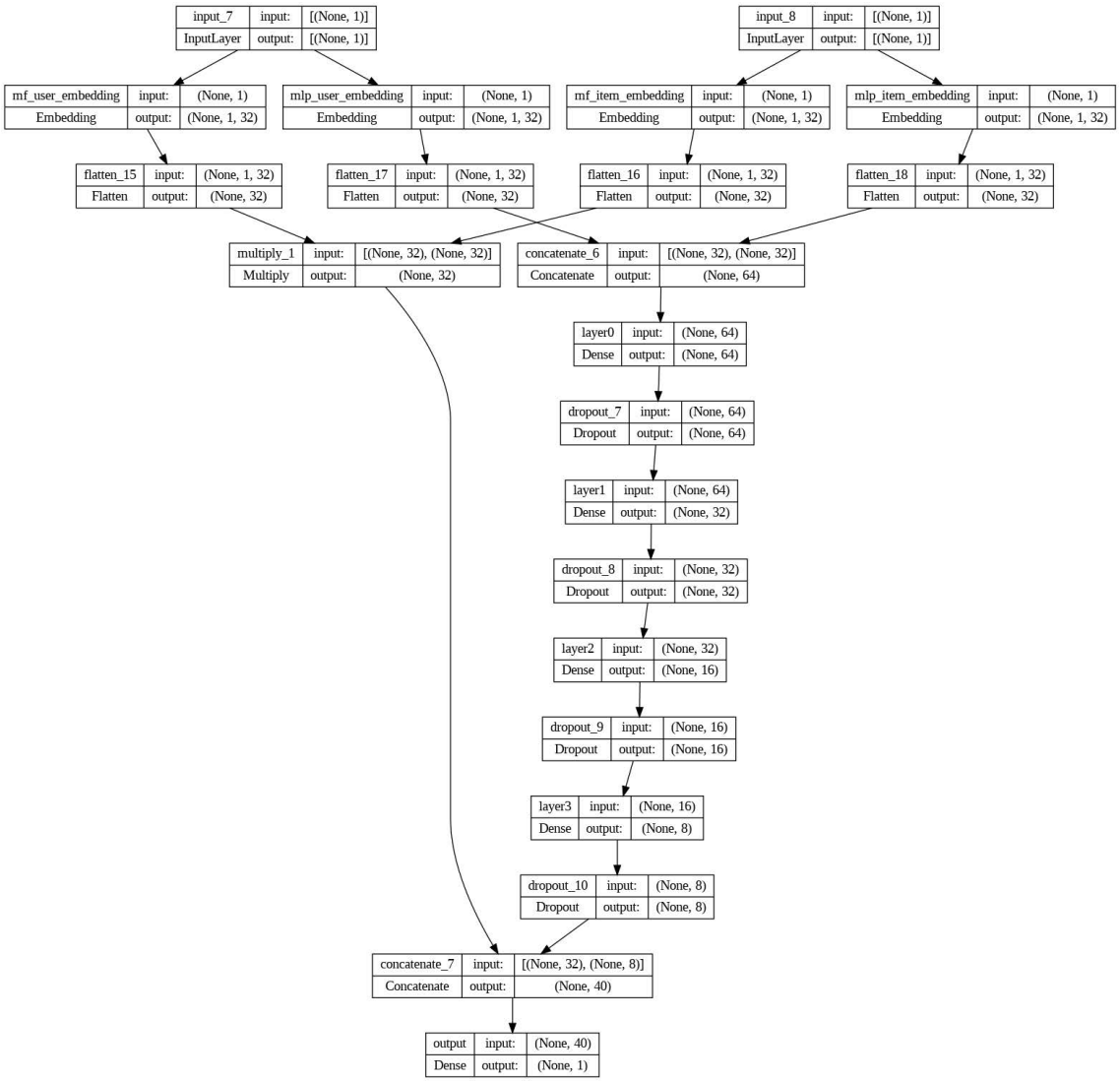

Figure 5.3: Architecture of NCF model with multilayer perceptron (MLP) and general matrix factorization (GMF) combined and represented with input and output shape of each layer.

size, 40). The final output layer will be a dense layer where the (batch size, 40) is converted to (batch size, 1) of 2D shape. The NCF architecture started with 3D inputs from the user and item embedding representations and was later converted to two-dimensional outputs until the final layer. The activation function used is ReLU in the MLP layers, and the sigmoid function is used in the final layer to converge the output with a range of 0,1. The kernel regularizer and initializer are l2 and lecun uniform, respectively.

Dimensionality reduction techniques	Parameters	Values
t-SNE	perplexity	50
t-SNE	random_state	12
t-SNE	n_components	2
t-SNE	early_exaggeration	12.0 (default)
t-SNE	learning__rate	100
PCA	random_state	12
PCA	n_components	2

Table 5.4: Dimensionality reduction methods PCA and t-SNE hyperparameters applied to the NCF results in each layer

Hyperparameters	Values
Latent dimension	8,16,32,64
Drop out	0.1,0.01,0.001,0.02,0.002
Regularization Value	0.1,0.01,0.001,0.0001,0.002
Learning Rate	0.001,0.0020,0.0040,0.0080
Batch Size	128
Activation Function	ReLu in MLP and Sigmoid in Concatenated layer

Table 5.5: Hyperparameters of the NLP model with the value choices for each run and collected respective output to form the unique values for each combination.

5.5 Visualization Toolbox

5.5.1 Visualization Toolbox Data

The data for the visualization toolbox are derived after applying dimensionality reduction techniques like t-SNE and PCA. Then collected the data as JSON and CSV files and used D3.js on top of it for visualizing the data. The dimensionality reduction methods have parameters that need to be provided for better results. The parameters used for t-SNE and PCA are mentioned in the table 5.4. After applying dimensionality reduction techniques like PCA or t-SNE, the high dimensional data of the final layers of the GMF, MLP, and NCF layers will be converted from 2511 (random sampled) values of the output layer of each GMF, MLP, and NCF layer to 2-dimensional data and then processed to form patterns out of it. Each output value from the output layer will be converted to 2-D data points.

For the hyperparameter visualization, the hyperparameter values are derived from the combination of the latent dimension, dropout, regularization value, learning rate, and batch size provided to the NCF model and to form the final RMSE values. Hyperparameters values are provided in the table 5.5.

Frequent itemset mining results are derived, and on top of the rules, the network diagram is built to show the relationship by fine-tuning the model with minimum support, lift, and confidence of 0.001, 1, and 0.5, respectively.

5.5.2 Visualization Toolbox Features

The visualization toolbox is basically built to find the relationship in existing customer-product interactions and model interpretability. For this purpose, we built the data mining model of frequent itemset mining and item-item collaborative filtering and then used the neural collaborative filtering model to find user-item interactions that were learned by the model. With all this data, how can we visualize it in order to get an overall understanding of the underlying data. The toolbox is built using Python for the backend with flask, javascript library D3.js (Data Driven Document), HTML, and CSS. D3.js [BOH11], is primarily chosen because it generates SVG (Scalable Vector Graphics) which seamlessly renders in all web browsers, including Microsoft Edge,Brave, Google Chrome, and all major web browsers, and it produces high quality and visually meaningful graphs. This also provides interactivity possibilities where the users can interact, and according to the interaction, the SVG element is adjusted and visualized. The Python flask application serves as a backend system that serves the request from the user and processes the respective HTML pages that need to be displayed on the front end with the help of the URL routings and static images preloaded for the HTML page. As per the request, URL routings match the HTML pages of different functionalities. The available views from the URL view and HTML are listed below:

- /homepage homepage.html
  - /internal internal.html
    - /internal - indexer.html
    - /options - options.html
    - /arresults - div_results.html
    - /ncfhyperparameter - ncf_hypervalue.html
    - /ncfresults - NCF_Results.html
    - /prodhier - product_hierarchy.html
    - /ncfcluster - ncf_cluster.html
    - /hiddenlayer - hiddenlayer_ncf.html
    - /combinedview - combined_view.html
  - /salesrep sales.html
    - /arrecommendation - div_recommend.html

Once the flask application is started, the front end of the application, as in fig 5.4 (1), will be displayed with the two buttons separating the sales representative view and the internal user view. From here, the total set of available views will be displayed according to the choices. The HTML page homepage.html will be allowing to select the respective views as per the button clicked by the user. Based on the selection of Internal view or SalesRep view, the user will be redirected to either fig 5.4 (2) or fig 5.4 (3) respectively. After this, the user can interact to choose from the options that are displayed. For instance, if the user selects the Association Rules Analysis, it will render the next page with options.html for choosing the respective customer division. By choosing the customer division, the respective div_results.html will be displayed by fetching the corresponding

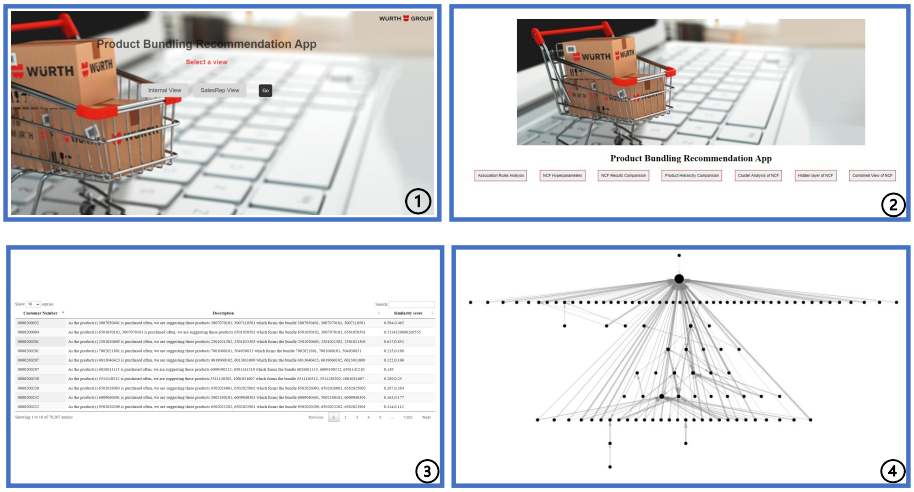

Figure 5.4: Available visualizations in the visualization tool.**(1):** Visualization tool homepage for having two separate views; **(2):** Visualization toolbox internal view, visualizing the product bundling results, relationship between the user and items, and model's interpretation;**(3):** Sales representative view of associative rules and item-item similarity in table view;**(4):** Association rules are displayed with the edges and nodes of the products that have different strengths and connectivities, portraying the product relationship.

input files and processing them. The option is taken care by flask application with the HTTP POST method. With the selected option to view the respective HTML pages, internal.html takes care of redirecting the request to render the specific HTML pages. The tool is designed to have the following visualization features:

- Network graph for representing the interactions of the sales data.
- Product hierarchical overview of the sales values of existing products in each level and customer division.
- NCF recommendation compared with actual products rendered as a collapsible tree with D3.js for tracking the whole product hierarchy.
- NCF hyperparameters parallel coordinates plot to have an overview of hyperparameters values.
- Lower-dimension projection of GMF, MLP, and NCF layers to analyze the patterns.
- Lower-dimensional projection with a scatter plot of GMF, MLP, and NCF layers combined in a single view.
- Heatmap of hidden layers from MLP and NCF Concatenated layers
- Results table view of association rules and item-item collaborative filtering.

Figure 5.4 (4) illustrates the connectivity of the products as the result of frequent itemset mining rules based on the purchased frequency and the relationship. Figure 5.5 (5) illustrates the hierarchical sales view of the existing purchasing data to represent the purchasing frequency at each hierarchical

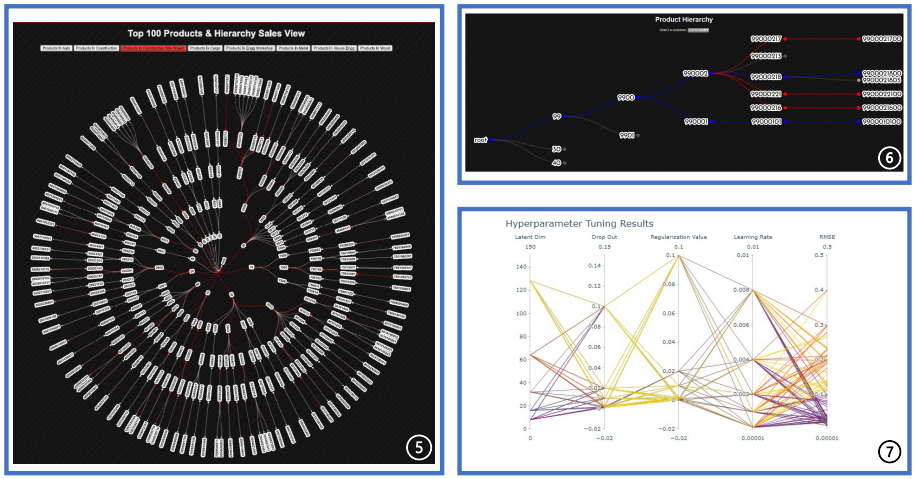

Figure 5.5: Available Visualizations in the Visualization tool.**(5):** Displaying the hierarchical overview of sales of the product with the color scale gradient to be high and low of sales value at each hierarchical level;**(6):** NCF results with red and blue for actual and predicted products, respectively;**(7):** Neural Collaborative Filtering with different hyperparameters evaluates the latent dimension,drop-out regularization value,learning rate,batch size, and the resultant RMSE values accordingly.

level. Figure 5.5 (6) represents the Collapsible Tree of hierarchical level view, with red and blue for actual and recommended products by NCF, respectively. Figure 5.5 (7) represents the hyperparameter representation of the NCF model. Figure 5.6 (8) projects NCF results analysis with dimensionality reduction techniques tSNE and PCA on the results of different components like MultiLayer Perceptron (MLP), General Matrix Factorization (GMF), and NeuMF combination of NMF and GMF. Figure 5.6 (9) represents the combined view of GMF, MLP, and NCF in a single view. Figure 5.6 (10) projects the heatmap representation of the hidden layer of each MLP layer and finally the concatenation layer in NCF. Figures 5.4 (3), 5.4 (4),5.5 (7),5.6 (8), 5.6 (9) and 5.6 (10) are displayed with a white background, and the remaining figures 5.5 (5), and 5.5 (6) are displayed with a black background because the white background made the nodes packed and hard to view the node values.

5.6 Visualization Toolbox Evaluation - Expert Study

For the evaluation part, the visualization tool is shown to both groups individually and evaluated with the requirements stated in the pilot study to be solved and the feedback for the results that are derived from the tool as an expert study. As the authors of [SMM12] mention about the implementation stage of the core phase, the visualization toolbox is evaluated by an agile way of improving each outcome shown to the domain experts with the suggestions. So, the evaluation is conducted as the expert study with the domain experts. Here, we consider group 1 to be the two technical people from the data science team, and group 2 represents the non-technical manager of the data science team. Some of the changes mentioned by the group 2 business users regarding the output visualization are to add table filtering properties. Earlier, it was displayed without any filtering capabilities, and the user and the products were recommended without the score that

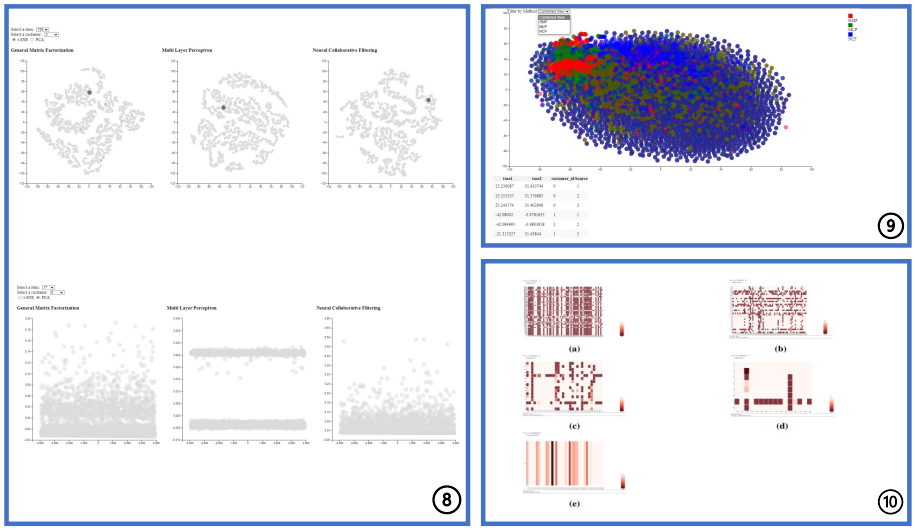

Figure 5.6: Available Visualizations in the Visualization tool.**(8):** NCF results analysis with dimensionality reduction techniques on the results of different components like MultiLayer Perceptron (MLP), General Matrix Factorization (GMF) and, NeuMF combination of NMF and GMF t-SNE and PCA reduction technique results; **(9):** NCF results analysis with dimensionality reduction technique tSNE to be in one combined view of MLP, GMF and, NeuMF combination of NMF and GMF;**(10):** plots of hidden layers derived from Neural Collaborative filterings's MLP part and finally display the concatenated layer output from both MLP and GMF layers' output (a): Hidden layer 1 of MLP, (b): Hidden layer 2 of MLP, (c): Hidden layer 3 of MLP, (d): Hidden layer 1 of MLP, (e): Concatenated layer of NCF.

determines the product bundle. Later, it was adapted with the mentioned changes of having the filter over the table and the score provided in a separate column mentioning the percentage accuracy that the product bundle will work out.

In the early stages, the product bundling was considered to be made only with Neural Collaborative Filtering (NCF) results, but due to the lack of explainability, we changed the product bundling results to take results from Frequent Itemset Mining (FQM). Taking this as feedback, we further focused on providing interpretability for the chosen model. Evaluated the model in terms of interpretability.

The final results of the visualization tool (Figures. 5.4, 5.5, and 5.6) are presented with the changes mentioned by the domain experts (groups 1 and 2) and analyzed to see how they can be leveraged in understanding recommender systems operation under the hood, especially algorithms that deal with deep learning, which often deals with explainability issues because of the lack of explainability and interpretability (black-box model). The outcome of the expert study considers the factors that are explained by the authors of [MD20]. We are considering certain factors that are applicable in our case. Resonant factors mention transferability and evocativeness in our context, showing how the current tool can be used by another person who can make it adaptable to future works. In our visualization tool, we can make it reusable with the help of uploading links from GitHub or any other common shared space, which can be taken as input, and then the results can be customized. For instance,with the Figure 5.6 **(8)**, we can provide data that is already clustered, change the clustered

data to be displayed, and point the source link to the source file. Then the tool works as intended to show the relationship of the uploaded data. Another factor considered is abundance, which makes the data used for achieving meaningful interpretations clear. As for the business view, the data represented as hierarchical edge bundling and a collapsible tree graph were easy to understand and able to provide insights from the data. As it deals with the whole product hierarchy, one can get more insights from the hierarchy's viewpoints. From this visualization, the business manager from Group 2 was satisfied with the understandability and business usage. From the remaining views, the graphical view was more about understanding the level of correlated products, i.e., which products have a stronger relationship with each other; this plays a really important role in understanding the product bundling process of products that are relevant to be considered. The remaining views with the neural collaborative filtering model seem to be more visually useful but not more insightful for group 2, but with the explanation, we can achieve the real need of the tool. Suggestions were made to prepare technical documentation for the use of the tool.

As with the data science technical team (group 1), the outcome is totally satisfied with the changes mentioned in the agile meetings and included in the tooling. As with the graphical view, Figure 5.4 **(4)** of the relationship gave more insights to the group, showing that they could visually as well control the tool with the functionalities to check the strength of the related products and then filter certain products to check with the whole graph view. With the success metric calculation with KPI, the representation of products suggested for each customer in each hierarchical level of view is considered, which satisfies the requirement of giving an overview of actual and recommended products to customers in a hierarchical view, Figure 5.5 **(6)**. The hierarchical edge bundling, Figure 5.5 **(5)** provides KPIs in terms of the hierarchical distribution of sales value in a single view, which would have been more difficult to analyze in a table or different visualization. This satisfies the requirement mentioned in the pilot interview for KPI calculations. This makes the resonant rigor[MD20] measure for the usage of the tool by others to get insights or to develop it further, which also indicates transferring the knowledge of grained facts to use the design for their respective purposes.

With the results from Figure 5.6 and Figure 5.5 **(7)**, the requirement of deriving the insights from the neural collaborative filtering is achieved. Figure 5.5 **(7)** displays the hyperparameters role in deciding the final results to be efficient with different parameters fine-tuned, trained, and then evaluated. Figure 5.6 **(8)** provides a comprehensive view in analyzing the pattern that the model learned and how the customers are suggested with the products within the group. This clearly achieves the goal of explaining why the customers are getting the product suggestions and the customers who influence these results for other customers, as NCF recommends products to a customer based on other customers similarities. Further insights provided the internal workings of the NCF model, which was quite intuitive for the data science team (group 1) as they got to know how the internals of the NCF model work. Figure 5.6 **(9)** and the weights that contribute to each layer Figure 5.6 **(10)**. A suggestion to improve is to mask the customer number and product number, which were taken care of in Figure 5.6, where each customer and product were numbered randomly.

6 Results and Discussion

This chapter deals with the workflow and functionalities of the visual analytics tool. We discuss more about how the tool works and the visualizations in it. Also, we will discuss the outcomes that resulted from the tool and the generic research on the topic of model interpretability. The results are evaluated with the data prepared as mentioned in section 5.3 and the functionalities are evaluated according to the requirements derived from the pilot study. All visualizations are illustrated using screenshots from the running version of the visualization tool.

6.1 Dataflow and Workflow of the Visualization Tool

6.1.1 Dataflow of the Tool

In this section, we will explain how the visualization tool functions, including the transfer of data based on front-end requests, the displayed content based on user selections, and the presentation of the information. Figure 6.1, represents the workflow between the components providing the data and the control flow for the visualization tool. When the main Python web application gets started, the flask web server gets started. Based on the interaction from the users on the front end, the data flow and corresponding visualizations will be displayed. As in the figure 6.1, once the tool gets started, the input from the user will be taken, the referenced data for the particular workflow will be retrieved, and then the respective HTML template file with D3.js or other libraries for the visualization will be rendered. The interaction from the user will trigger the POST HTTP method. The flask application will handle this POST request by having different URL routings, which is the driving factor for the HTML pages. Based on the request, the input data varies; it can be JSON or CSV files. So, the request passes either a JSON or CSV object to the respective HTML page. Visualization is mainly done with the help of D3.js, networkx library for network graph, and plotly for the parallel coordinate plots. D3.js reads the processed object and renders the visualization as per the request passed on by the user. The homepage.html file provides two options for the internal users (internal.html) and sales representative (sales.html) views. From here, the views are further divided into their respective functionalities. The internal.html file proceeds further with the various HTML files for the interpretability of the NCF model and results, the network graph for product bundling, and the internal KPI sales hierarchical view. The files indexer.html and options.html are the options pages displayed with dropdowns to choose from for the selection of customer division, respectively. Since the view depends on the customer's selection from eight different divisions, namely Auto, ConstructionSite Project, Construction, Cargo, Metal, Wood, House Engineering, and Engineering Workshop (Betriebswerkstatt), the options.html file takes care of this dropdown list. The remaining html files like combined_view.html,

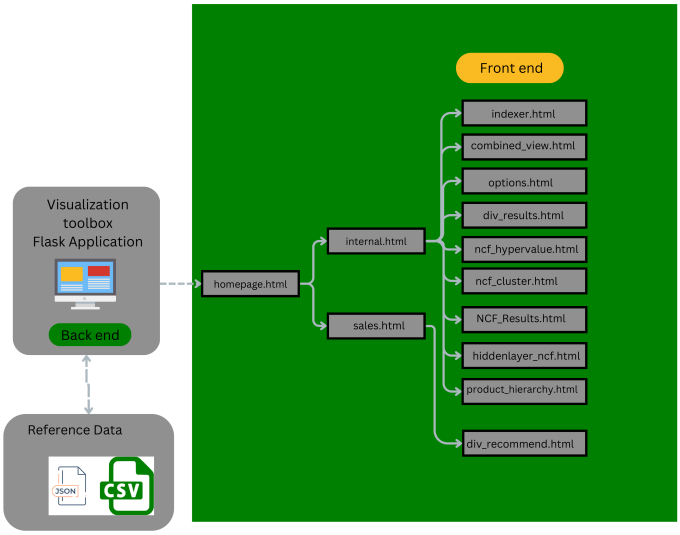

Figure 6.1: The workflow of the visualization toolbox represents the front-end and back-end components, and the data flow between the components. The short dashed line represents the control flow, and the long dashed line represents the data flow between components. The normal straight lines represent the relationship between the components.

div_results.html, ncf_hypervalue.html, ncf_cluster.html, ncf_results.html, hiddenlayer_ncf.html, product_hierarchy.html, and div_recommend.html are all dynamic files that react based on the users' choices.

6.1.2 Workflow of the Tool

Once the visualization tool is started, it will be displayed with the option to choose either internal view or salesrep view (Fig 6.2 (1)). Based on the choice, the internal view will be displayed for the data science team (Group 1) with the options of association rule analysis, NCF hyperparameters, NCF results comparison, Product Hierarchy comparison, cluster analysis of NCF, hidden layer of NCF, and combined view of NCF (Fig 6.2 (2)). We will explain each functionality in further sections. On the contrary, if we select the salesRep view, the table will be displayed as in Figure 6.3 (2) with the customer number, a description of the product bundle details, and their respective similarity score. If it has a single score, then it is the result of associative rules, and if it has multiple scores, then it is the result of item-item collaborative filtering. The table will be displayed based on the choices from the dropdown of customer divisions (Fig 6.3 (1)). The table is interactive, we can filter based on particular customers or products. It can be configured with the number of rows to be displayed in the table, sortable by customer number, and similarity scores from the table.

In the internal view, we will be displayed with options to choose from, as in Fig 6.2 (2). The first option, association rule analysis, will be provided about the product relationship derived based on the customers' purchase history. In this view, we can select the customer division as in Fig 6.3 (1). After selecting the corresponding customer area, the graph will be displayed. So, in total, eight

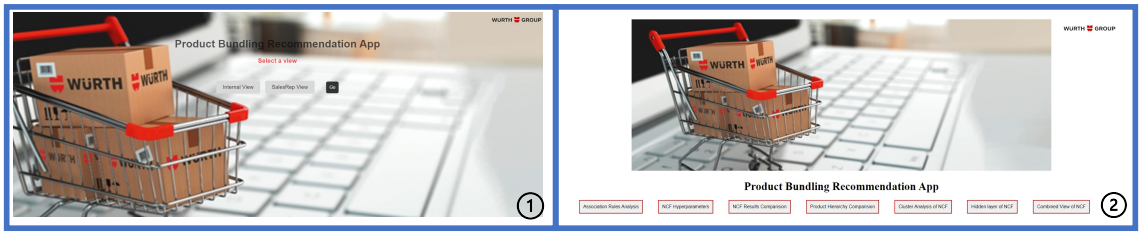

Figure 6.2: (1). Initial page display with two views Internal and SalesRep view, (2). Internal view with NCF model interpretability options.

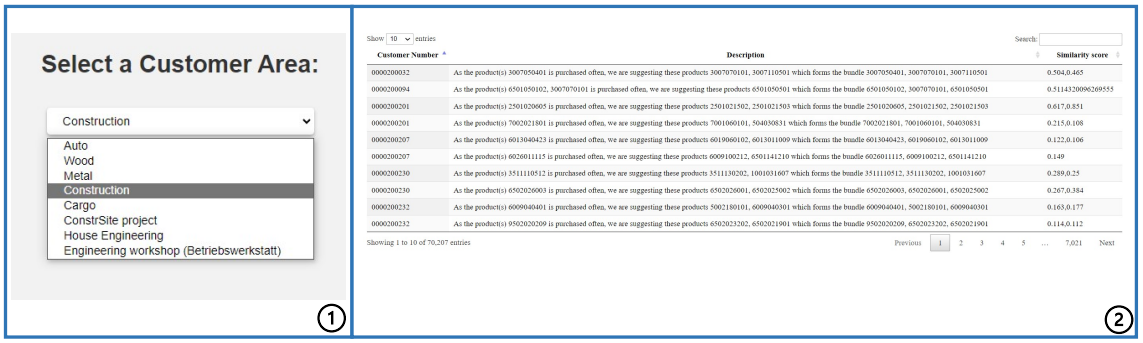

Figure 6.3: (1). Customer divisions dropdown with eight choices, namely Auto, ConstructionSite Project, Construction, Cargo, Metal, Wood, House Engineering, and Engineering Workshop (Betriebswerkstatt), (2). SalesRep table view of the product bundles along with the score percentage.

graphs will be displayed based on the selection, as we have eight customer divisions(Figure 5.4 (4)). The products will be displayed according to the strength of their similarity with each other, showing the relationship between the products.

Further from the internal view, if the user selects the product hierarchy comparison option, it will lead to the hierarchical edge bundling view (Figure 5.5 (5)) from the root to the child (six-level) hierarchy view of the products. At each level of the hierarchy, the color coding is encoded to represent the sales value of each product at its hierarchical level. This level deals with the top 100 products data, and the default view will display the top 100 products for all customer divisions. It can be further filtered by buttons for different customer division options, and the corresponding top 100 products will be displayed.

In the internal view, if the user selects the NCF results comparison option, it will lead to the collapsible tree view (Figure 5.5(6)) from the root to the six-level hierarchical view of the products, with the actual products being in red color and the predicted products being in blue color. If the user selects the NCF hyperparameters option from the internal view, it will lead to the parallel coordinate plot view (Figure 6.4) of the Neural Collaborative Filtering with different hyperparameter evaluations with latent dimension, dropout, regularization value, learning rate, and the resultant RMSE values accordingly. The user can drag axes and change the level of view that can be visualized. The values of the axes are decided based on the values for each hyperparameter. Making the axes have uniform values will make them difficult to interpret or make the analysis cluttered. The parallel plot also has the option of highlighting certain values from each axis and applying

Latent dimensions	Colors
8	Purple
16	Violet
32	Brown
64	Light Orange
128	Gold

Table 6.1: Color of parallel coordinates plot lines decided by latent dimensions.

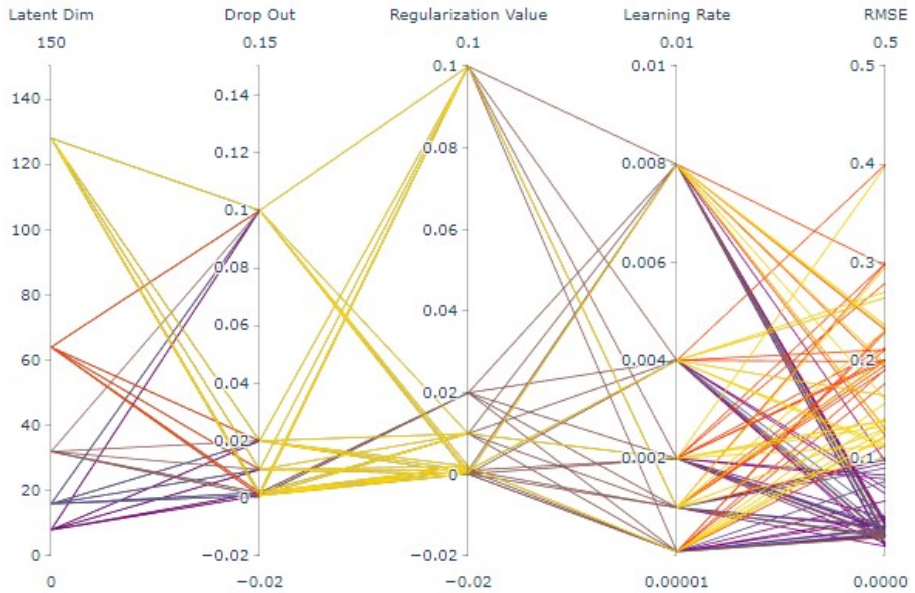

Figure 6.4: Neural Collaborative Filtering with different hyperparameters, evaluation with (left to right) latent dimension, dropout, regularization value, learning rate, and the resultant RMSE values accordingly. Color coding is mentioned in the table 6.1.

different color coding to the results, which will allow for more specific analysis to be done. We can select a certain value from the axes, and based on that, the following values in the axes will be highlighted, which gives a clear overview of the results. Figure 6.4, depicts the overall performance of all trained models with different hyperparameters and final RMSE error values.

If the user selects the cluster analysis of NCF, the figure 5.6 (8) will be displayed. It is a representation of the NCF, GMF, and MLP component output layers. To have an overview of how each model learns and patterns are formed. For this purpose, we used dimensionality reduction techniques PCA and t-SNE and reduced the high dimensional values to 2D values, which we then represented as a scatter plot in the graph (Figure 6.5). This helps to compare the patterns learned by each model in their respective training. Further, we can select the product to view the intensity of the predicted product for each customer. Also, to select a particular customer, we can view exactly where the customer belongs in the 2D representation. And we can drag and drop a scaled view for the purpose of selecting a certain area of customers and viewing the respective t-SNE values and the customers present in that selection.

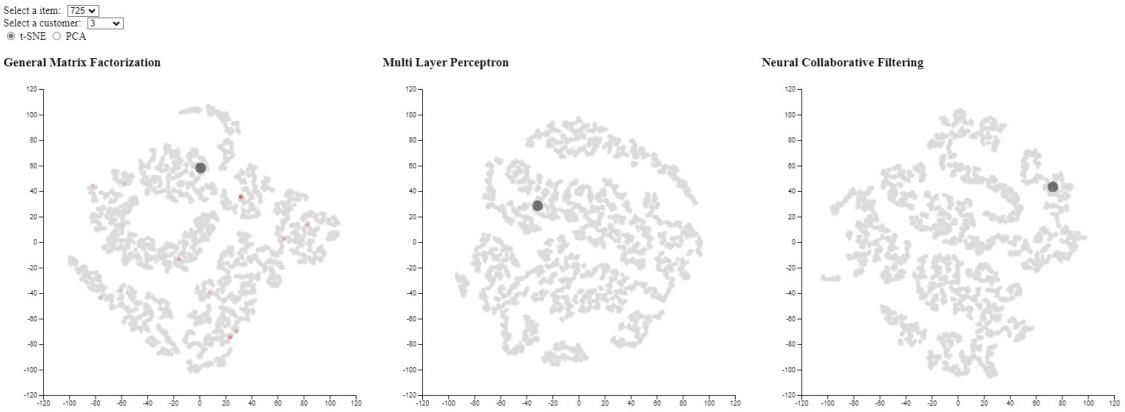

Figure 6.5: NCF results analysis with t-SNE dimensionality reduction technique on the final layer results of MultiLayer Perceptron (MLP), General Matrix Factorization (GMF), and NeuMF combination of NMF and GMF.

If the user selects the combined view of NCF, the figure 5.6 (9) will be displayed. It's the representation of the output layers from the NCF, GMF, and MLP parts in a single view to have an overview of how each model learns and the patterns that are formed. For this purpose, we used dimensionality reduction techniques like PCA and t-SNE and reduced the high-dimensional values to 2D values and represented them as scatter plots in the graph, differing from the above clustering graph by combining the values from each output in a single data frame and then applying the dimensionality reduction method. Further, we can select the respective model from the dropdown to see how the model is learned with the underlying pattern of the other two models. From the graph, we can drag, similarly to in the NCF cluster analysis, to check the users and their respective t-SNE values and then compare the values from the three models.

If the user selects the hidden layers of NCF, the figure 5.6 (10) will be displayed. Here, we could get the overall hidden layer weight distribution. Figure 5.6 (10) (a), (b), (c), and (d) depict the hidden layer distributions of each hidden layer 1, 2, 3, and 4, respectively. Figure 5.6 (10)(e) depicts the final concatenated layer of NCF, with GMF and MLP being passed on to the final hidden layer. This gives an overview of how the weights are learned with each hidden layer and which values are providing more impact in that corresponding layer. The values are matched with the respective colors of the red scale from minimum to maximum.

6.2 Discussion and Analysis

In this section, we will learn about the analyses that can be driven from our visualization tool and how it solved the research intent regarding the problem we took on. First, the interpretation of the existing product relationships from the sales data will be analyzed, which will be the more important use case for product bundling. Second, we will look at how we could leverage the KPI views with the help of visualization. Third, we analyze the pattern that is formed from each model separated and interpreted with dimensionality reduction techniques. Fourth, we will look at the neural collaborative filtering results influenced by the hyperparameter values. Fifth, we will analyze the combined view of the NCF, GMF, and MLP models and find the learning from the model. Sixth,

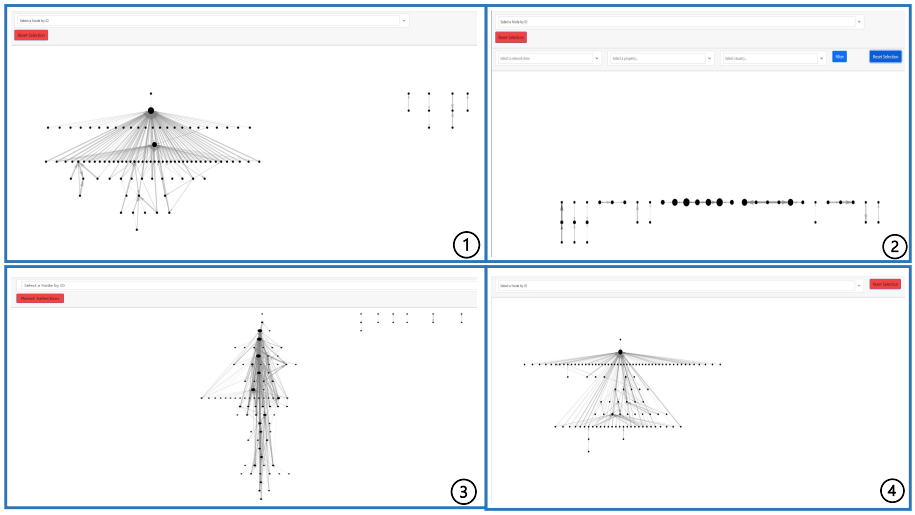

Figure 6.6: Network graph of association rules derived from the frequent itemset mining algorithm. (1). represents the associative rules for the auto customer division; (2). represents the associative rules for the wood customer division; (3). represents the associative rules for the metal customer division; (4). represents the associative rules for the construction customer division.

we will check how could we visualize the hidden layers of the MLP and the concatenation layer as heatmap. Lastly, we will analyze how to represent the matrix with the help of matrix bandwidth minimization techniques and the outcomes from them.

6.2.1 Analysis 1: Interpretation of the Existing Product Relationships

The main goal of this thesis is to derive product bundles for the purpose of recommending products as bundles to the customer. For this purpose, we have developed a finely tuned frequent itemset mining algorithm. This results in the products being purchased together at the base level, as, like rules, the antecedent will lead to the consequent. The base-level working of the algorithm is clearly defined in the Methods chapter, frequent itemset mining section 5.1. After applying frequent itemset mining to the user-item interaction dataset, we derive insights from this algorithm with the help of a network graph. And further, as we are dealing with the customer-oriented bundle suggestion, we take the results from the associative rules, compare the purchasing behavior of the customer, map the product bundles that are more similar to the existing purchasing behavior using the Jaccard similarity index, and then suggest the product bundles to each customer according to the match rate and the confidence score. But before doing the similarity score calculation, we need to get insights into the existing product relationships, which will be more relevant to suggest, and which products are often purchased together and the correlation between them from the results of the associative rules. The network graph will be visualized for each customer division based on the selection from the dropdown (Fig:6.3 (1)). The results from the associative rules are displayed in the Figures 6.6 and 6.7.

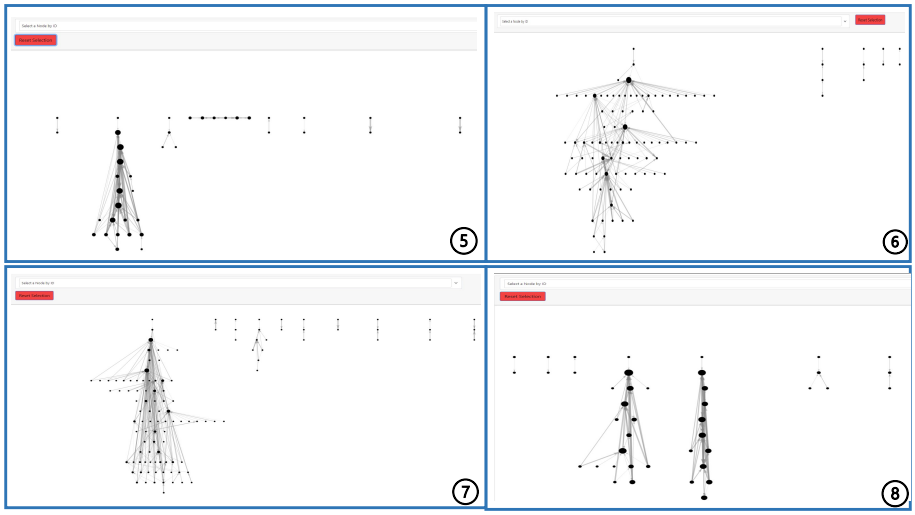

Figure 6.7: Network graph of association rules derived from the frequent itemset mining algorithm. (5). represents the associative rules for the cargo customer division; (6). represents the associative rules for the construction site project customer division; (7). represents the associative rules for house engineering customer division; (8). represents the associative rules for the engineering workshop (Betriebswerkstatt) customer division.

(a) **(b)**

Figure 6.8: Association rules derived from the frequent itemset mining algorithm for the construction customer division. In (a) a particular node selected with a mouse click is highlighted and for (b) a node is selected from the drop-down filter and highlighted.

As we can see from the graph, the nodes are interconnected with other nodes through directed edges. Here, the nodes are the products, and the edge represents the confidence score, which means how much those two products are correlated with each other, which makes those products bundle together. We here take the construction customer division (Fig:6.8) to analyze the results. To provide more insights, we have added the option to hover over the node to see the product names, and on selecting the product, the nodes get highlighted and the corresponding edges are highlighted. As in figure 6.8(b), a product number of 1004012208 is chosen from the dropdown, and the corresponding node with that product number gets highlighted, and we can see the connectivities to other products in the graph. Here, the product: "Winkelverbinder Typ A" is further connected to two other products, namely, "ASSY 4 WH Tellerkopfschraube Stahl verzinkt teilgewinde Scheibenkopf" and "ASSY 4 CSMP Universalschraube Stahl verzinkt Teigewinde Senkfrastaschenkopf". The connectivity

between the two product nodes is represented by the thickness of how they are correlated. If we compare the remaining nodes, we can see the difference in thickness of the edges connecting other products. Also, the products connected to a single product will be further connected to other products. From this, we can derive the relationship that arises from a single product and how other products are connected with the correlation. This correlation will help to derive new products from the connectivities between the products. By examining the relationships among products, we can extract association rules, and utilize network graph visualization to create new product bundles. In this case, a product is chosen from the dropdown list, and the corresponding connected nodes are represented with the connecting edge values from the association rule discovery. With these results, we could further see the connections to the other nodes, as the connected node with the chosen product node will have associations with other nodes that are not directly connected with the chosen product. From this, we can derive new products that are correlated, and we can form product bundles out of this relationship. Further, we can drill down to the bottom level. By doing this, we can increase the number of products in the bundle and also filter out the ones that don't have a strong correlation with each other. So, from this network graph representation, we can derive the existing product relationships, further bundle the products together, find new relationships between the products, and form new bundles.

6.2.2 Analysis 2: Visualization of hierarchical sales views and NCF results

As per the requirement from the pilot interview, to display the hierarchical view of the sales structure in a feasible way to understand the products' sales in each hierarchical level to provide as a KPI to understand the sales in a much better and more understandable way. In the current structure of the product hierarchy, there are five levels: product group, product class, product family, product model, and product. It is in the order of highest to lowest hierarchy level. So, to visually see all those data sets in detail, it would be harder with the table view or dashboard. For this purpose, we have used hierarchical edge bundling because it helps to visualize adjacency relations between entities organized in a hierarchy. This works by bundling the adjacency edges together to decrease the cluttered view of an enormous amount of data. In the figure 5.5 (5), we could see the hierarchical overview of sales of the product with the color scale gradient to be high and low of sales value at each hierarchical level, and this is considered with the overall customer divisions. With the options above for eight different customer divisions, we can drill down to each of them and see their respective sales value in the hierarchical view. In the figure 6.9, we can see all of the hierarchical sales views for eight customer divisions. The color grading is done from the red color to the white color, with red being the maximum and white being the minimum. Figure 6.9 (a) represents the hierarchical view of the customer divisions Auto (left) and Construction (right); Figure 6.9 (b) represents the hierarchical view of the customer divisions ConstructionSite Project (left) and Cargo (right); Figure 6.9 (c) represents the hierarchical view of the customer divisions Engineering workshop (Betriebswerkstatt) (left) and Metal (right); Figure 6.9 (d) represents the hierarchical view of the customer divisions House Engineering (left) and Wood (right).We will take an example from customer division Auto and explain the insights that are driven from this view Figure 6.9 (a) (left), product group 75 is marked as medium red scale, and going forward, the sales value of its child product class is considered with the values 7501, 7506, and 7509. At this level, we could see the color variance happening because of the sales values split in the product class, where the product classes are color graded from maximum (red) to minimum (white) in the order of 7506, 7509, and 7501, respectively. In the next-level product family,the values are ordered as per the color

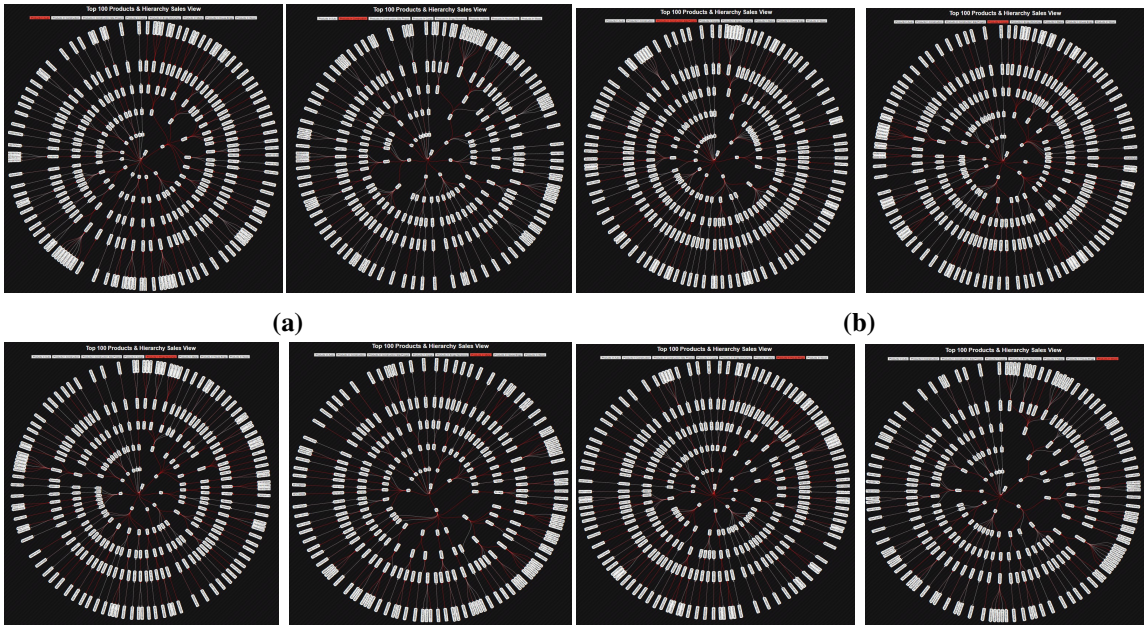

Figure 6.9: Hierarchical edge bundling results of the top 100 products for eight customer divisions with the sales level in color grading interpolated with low and high red colors, (a): results of customer division auto (left) and Construction (right); (b): results of customer division construction site project (left) and cargo (right); (c): results of customer division engineering workshop (Betriebswerkstatt) (left) and metal (right); (d): results of customer division house engineering (left) and wood (right).

grading in the order 75090202, 75010308, 75060802, 75060610, 75061104, and 75061122. Further in the product model, we will have the ordering as follows, 7506080202, 7501030801, 7509020205, 7506061030, 7506110404, and 7506090469. The ordering of each value depends on the sales value at each hierarchical level, not the particular hierarchy group. So, this hierarchical edge bundling gives a clear overview of how the sales value is distributed over the hierarchy.

With the next analysis, we can display the Neural Collaborative Filtering (NCF) with the same hierarchy level. But for the purpose of being more dynamic and not restricting the results, we adopt different visualizations. Since hierarchical edge bundling is good to display all the hierarchical levels of the whole customer division, here we are concerned about a particular customer, his purchased products, and the products that are getting recommended, not the whole product hierarchy. We have used a collapsible tree as the visualization representation since it collapses the hierarchy that is not getting displayed for a particular customer, which helps to visualize only the product hierarchies that a particular customer has already bought or is getting recommended. So, with the collapsible tree set to be dynamic, the values that are represented will be changed according to the customer being selected. From the figure 6.10, we can derive insights into how the actual product hierarchy levels (figure 6.10 (1)) are getting displayed (due to the size constraint, it was reduced to 3 product groups). When we select a particular customer, the tree levels get automatically shrunken down to the product group level and then displayed until the lowest level in the product hierarchy. And to find the products that are getting recommended to the customer, we color code the

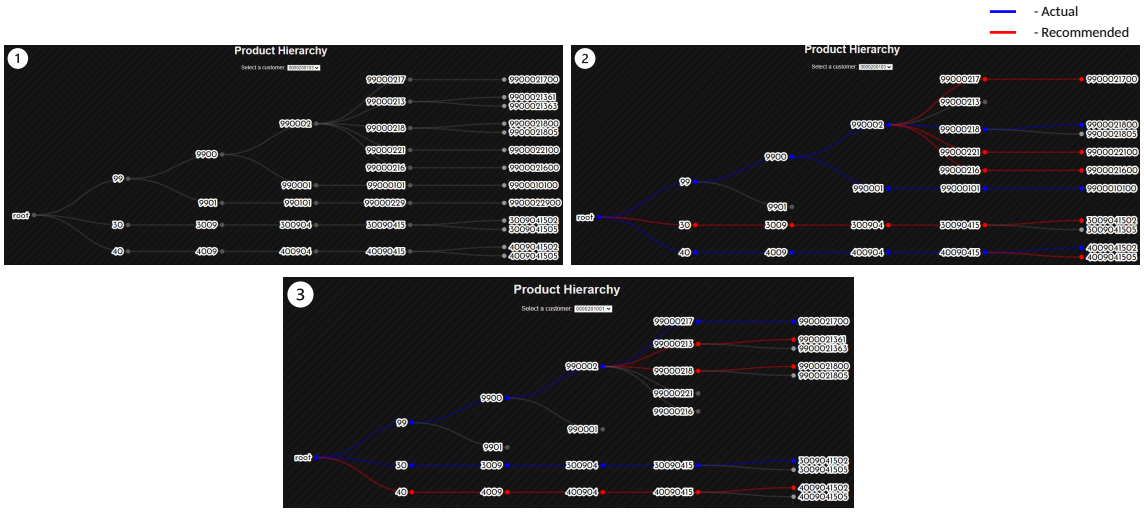

Figure 6.10: NCF models' results are displayed with the product hierarchical level (1). Default view of the overall hierarchical view of products represented in the collapsible tree; (2). Filtered view of the collapsible tree for a particular customer (0000200103) with the remaining edges that are collapsed; (3). Filtered view of the collapsible tree for a particular customer (0000201001) with the remaining edges that are collapsed.

connecting edges. If a product has already been purchased by the customer, we color code it blue, and if it's recommending a new product, the one that has not been purchased is represented with a red color. Here the customer 0000200103 is selected, and the products 9900021700, 9900022100, 9900021600, 3009041502, and 4009041505 that are already bought by this customer are color coded as blue till the root node, and the recommended products 4009041502, 9900022100, 9900010100, and 9900021800 are color coded as red till the root node. It is helpful to understand at which level new products are being recommended, and we can further analyze whether this level is relevant to the customer or not based on the heuristics derived.

6.2.3 Analysis 3: Pattern Learning of the NCF, GMF, and MLP models

This part deals with the understanding of the patterns learned by Neural Collaborative Filtering (NCF), MultiLayer Perceptron (MLP), and General Matrix Factorization (GMF). As in figure 6.5, we have three different views of the model pattern in the order of GMF, MLP, and NCF (from left to right), respectively. This is the outcome of the construction customer division model. We built the model separately for each customer division to reduce the size of the data to be trained with and the computational complexity. The construction metal division is trained with 15.8 million sample data points and tested with 3.9 million sample data points. As we have tested the model for hitrate and the NDCG evaluation metric as defined in section: 5.2, we got this training and testing sample data after the temporal split. We train the model with 30 epochs and a batch size of 128. For the purpose of illustration, we randomly take data for 7811 customers and then transform the output into a 2D view with the help of dimensionality reduction techniques. We apply dimensionality reduction to the results derived from the output layer of each NCF, GMF, and MLP model. The output will be derived from the last layer, as in the figure 4.7. The output that is produced from each of the models is in the shape of (7811, 2540). For each user, the output is whether the product is getting

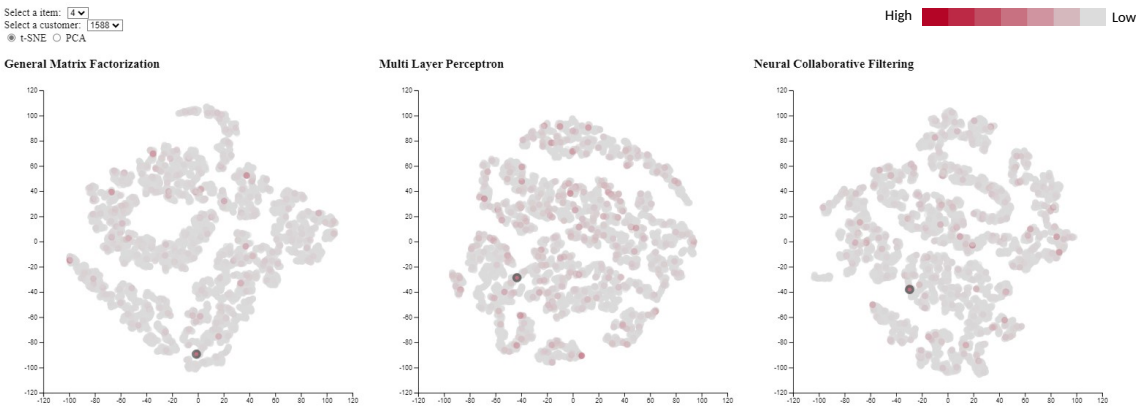

Figure 6.11: Pattern analysis using t-SNE dimensionality reduction for the product 35080703100 (encoded to 4) and customer 0000382835 (encoded to 1588) with the color coding provided in the scale below, ranging from high to low based on the values of the recommendation score.

recommended or not for the whole 2540 products. If it is recommended, then it will have a value in the range of (0, 1], and if it is not recommended, it will be zero. So, with the help of dimensionality reduction techniques, we reduce the value of 2540 features into two-dimensional features, resulting in the shape of (7811, 2). With this dimensionally reduced data, we can find the pattern for each user to be represented in a 2D view. So, this provides the data to be represented for each customer and also has an influence on the products' rating based on the recommendation score generated by each model. With this visualization (Fig: 6.11), we have the option to select a particular customer and see where the customer is placed in the three models' results by highlighting that point in the scatter plot. And we can select the nearby range of that customer and check to see if the products that are suggested to the customer are similar to those of the other customers in the range. And then, with the possibility of selecting a product and seeing the color interpolation based on the recommendation score that is produced by each model, it dynamically changes the color based on the minimum and maximum values for each product in their respective model results.

For a clearer understanding, we take a random customer and the products that are getting suggested for that customer and then derive an analysis on top of it. As per the figure 6.9(a)(right), the product 35080703100 is the most bought in the construction customer division, with the color grading of red at the maximum of the scale. And we will take the customer 0000382835 and analyze how the products are recommended for the customer and the insights driven from it.

As in the figure 6.11, with the help of t-SNE dimensionality reduction, we derive the pattern displayed. The customer 0000382835, who is encoded to 1588, is selected from the "select a customer" drop-down selection, and the corresponding customer is selected with the black circle enclosing the point of the customer number highlighted. The product 0000382835, which is encoded to 4, is selected from the "select an item" drop-down selection, and the corresponding product in the whole scatter plot in each model result will be highlighted separately in the color scale provided in the figure 6.11. The color scale is decided based on the value of the selected product and its recommendation score generated by each model, and then the color is assigned a value based on the maximum and minimum values taken as a range. So, the recommendation score from each model will have its scale, and then the customers who bought those products will be highlighted with the

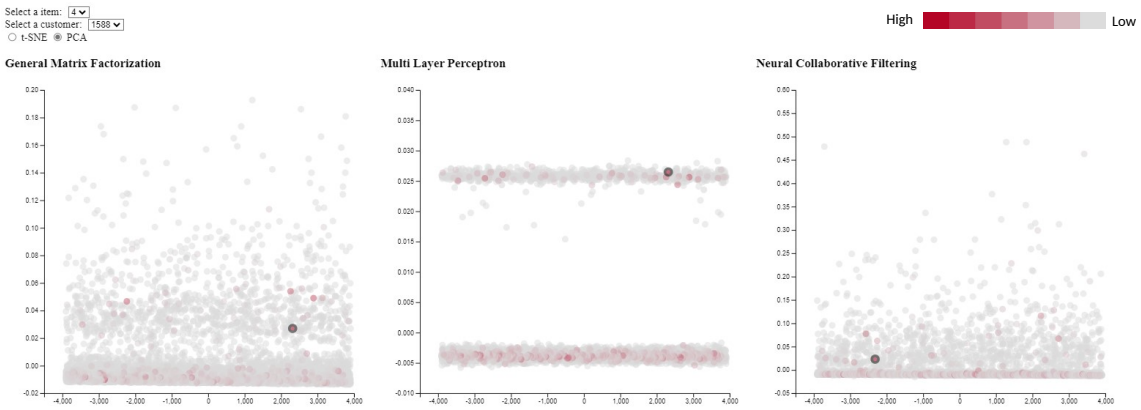

Figure 6.12: Pattern analysis using PCA dimensionality reduction for the product 35080703100 (encoded to 4) and customer 0000382835 (encoded to 1588) with the color coding provided in the scale below, ranging from high to low based on the values of the recommendation score.

color scaled value in the scatter plot of each model, respectively. From the figure 6.11, we can understand that the values of the NCF model are influenced by both the GMF and MLP models. We could see the values in the GMF and MLP are similar to the pattern of the values in the NCF; the value for the customer 0000382835 remains higher in all three scatter plots. And the value for the customer 0000382835, as seen in the MLP, is similar to the NCF results. This helps to draw the conclusion that the models' pattern learning of NCF has more influence from both the GMF and MLP. Irrespective of the particular customer, the values highlighted in the GMF and MLP are combined levels of view for the NCF. As we can see from the values in NCF for this particular item, the values are distributed as a combination of GMF and MLP. In NCF, the values are ranged from maximum to minimum, so the values are close to the minimum value, which is why more values are closer to gray than red. From this also, we can infer that the values won't be added up from MLP and GMF; they perform based on the group of customers present and the number of products suggested to the group of customers.

With the help of the PCA dimensionality reduction method, the figure 6.12 is derived with reduced dimension representation. Even in this case, we could see that there were a lot of recommendations provided to all the customers with the product 0000382835. This depicts that the product that is bought more will be suggested to the customers as per the regular recommendation. To be noted, these results are not filtered out for the products that have already been bought by the customers. We will filter them only for the final representation. Another interpretation is that with PCA, we cannot derive more information about the pattern because it is more cluttered in the representation and it exhibits linear pattern in the non-linear representation.

From the figure 6.11, the pattern that gets formed is analyzed here. The pattern of the GMFs' certain area at the top left side of the scatter plot is similar to the NCFs' top left side of the scatter plot. The patterns that are formed in MLP are influencing more of the patterns in NCF. As seen in the plot, the MLPs' middle area and the area below it are more similar to the NCFs' middle area and the area behind it.

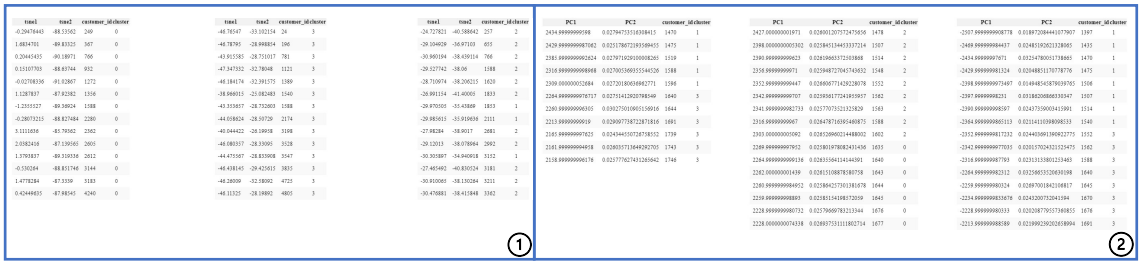

Figure 6.13: Table with the customers in closer proximity to the customer chosen, which is selected based on the free lasso selection (1). t-SNE dimensionality reduced values after selecting customer 0000382835 and their respective customer nearby groups. (2). PCA dimensionality reduced values after selecting customer 0000382835 and their respective nearby customer groups.

And another aspect to be analyzed here is that the group of customers who are recommended the same products will be in close proximity. We have tried applying K-means clustering to group the customers, but the clustering did not give proper clusters in this sparse data representation. So, we removed the clustered outputs from the displayed scatter graphs. With the help of hovering around a customer in the scatter plot, we can get the table with the customers who are similar to the customer chosen, and those points tend to occur closer to each other and have a strong influence on the products that were suggested to that particular customer. As in figures 6.13 (1) and (2), we could see the customers who are influential for the products that are getting suggested to the customer 0000382835 by the t_SNE values and PCA values, respectively, in the corresponding tables.

From these three analyses, we can derive the deep insights of the NCF model. From these figures 6.13 (1) and (2), we could understand how the products are getting recommended to the customer and the group of customers who are related to that group. We can derive insights from these figures 6.11 and 6.12, about how the products are getting recommended with the score and how well the customers are getting recommended with the products of high sales value. Also, we understood the pattern that was learned by the NCF model to be influenced by GMF and MLP values. This helps to understand that the NCF model is an ensemble learning model that learns the linear features of GMF and the non-linear features of MLP and together provides an output that is more important than the general matrix factorization learning of linear features. Also, tried different dimensionality reduction techniques like MDS and UMAP. MDS resulted in a more cluttered representation, which didn't help to get more insights from the output. With UMAP, the technique didn't work to handle the large sparse data, and the results were not produced for further analysis.

6.2.4 Analysis 4: Hyperparameter values interpretation for the NCF model

In this section, we will discuss the hyperparameter values interpretation of the NCF model. We have trained the model with the data that's been prepared and mentioned in the section 5.3. A parallel coordinate plot is used to visualize the various hyperparameter values and how they influence the final metric that is assessed for the models' performance. We can visualize the hyperparameter parallel coordinate plot in fig 6.4 and overall performance with the different values. It's a dynamic plot where the user can select a particular value on an axis and check the corresponding values that are getting driven from that initial selection. In the parallel plot, there are five axes, of which four

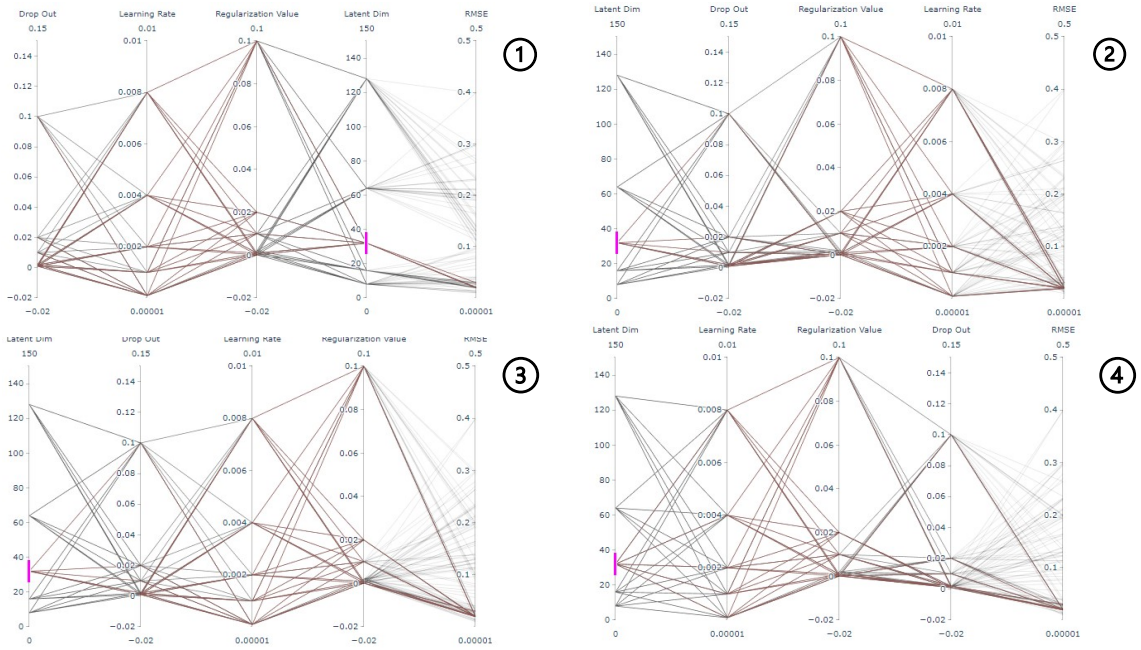

Figure 6.14: Parallel Coordinates Plot of the NCF model with latent dimension 32 (selection as pink) as value (highlighted in brown). (1): Normal placement of the hyperparameters latent dimension,drop out regularization value,learning rate, and the resultant RMSE; (2): Representation of learning rate at the second axes; (3): Representation of regularization value before the final axes; (4): Representation of dropout value before the final axes.

are the hyperparameter values, and the last axis is the output of the model, RMSE values. RMSE value is calculated to assess the performance of the model. So, we analyze the performance of the model with the values from the hyperparameters of the first four axes and track the respective change in RMSE value on the final axis. The order of the four hyperparameters placed on the axes is latent dimensions, dropout, regularization value, and learning rate, and finally, the resultant RMSE value is placed on the final axis (left to right). The colors of the parallel coordinate plot lines depend on the value of the latent dimension hyperparameter. The latent dimensions and the corresponding colors are represented in the table 6.1. Figures 6.14 and 6.15 represents the parallel coordinate plots of latent dimensions 32 and overall values for the latent dimensions respectively, with different hyperparameter values.

As in figure 6.4, when the NCF model is trained with 32 latent dimensions as input, the model performs better with an RMSE value of 0.012277, which is the lowest of all the hyperparameter combinations. This is the reason that we chose the latent dimension value to be 32 and fixed the model to get good, accurate values. Further, the values of this better performant model are influenced by the dropout setting of 0.010, the regularization value of 0.0001, and the learning rate of 0.008. The model is trained with activation functions: ReLU for the MLP layer part and sigmoid activation functions for the final concatenated layer output. Generally, the model should perform better when the latent dimensions are increased by a certain amount. On the contrary, in our case, the model performed well with latent dimensions of 32 compared to the remaining 64 and 128 latent dimension values. This result comes regardless of the remaining hyperparameters. Even after

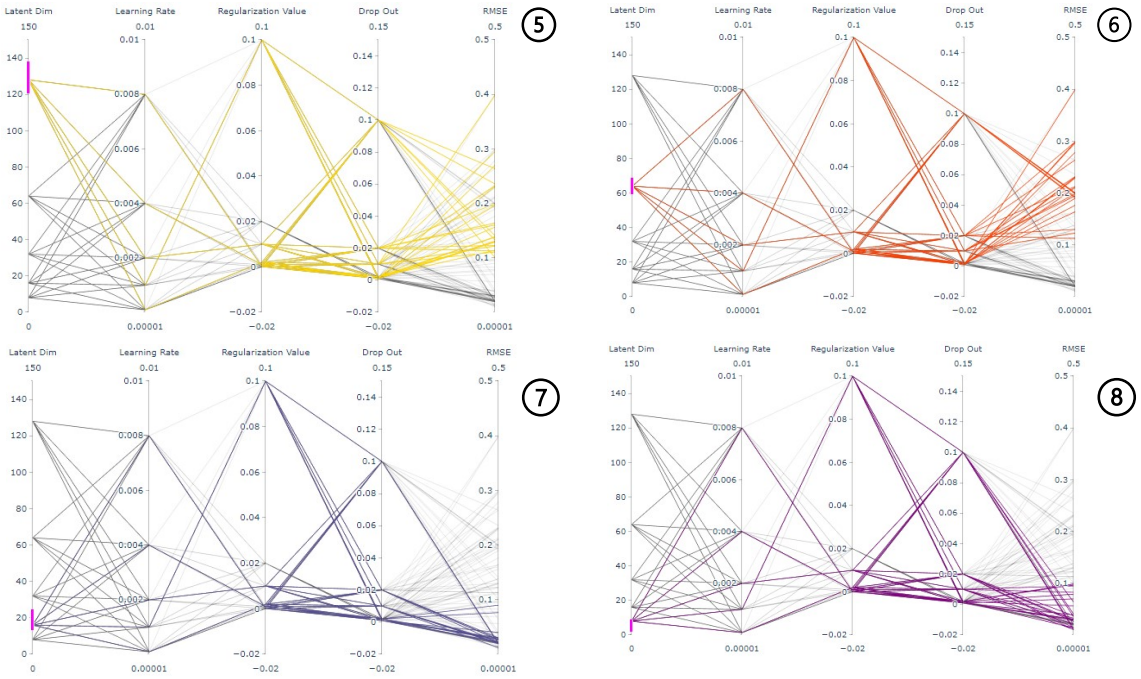

Figure 6.15: Parallel Coordinates Plot of the NCF model with (left to right) latent dimension,drop-out regularization value,learning rate, and the resultant RMSE values; (5): represents the value for latent dimension 128 (selection as pink); (6): represents the value for latent dimension 64 (selection as pink); (7): represents the value for latent dimension 16 (selection as pink); (8): represents the value for latent dimension 8 (selection as pink).

tuning the hyperparameters, the error values are not reduced. But as a whole, the error values do not deviate much, and they are still less than 0.5. When we talk about the RMSE value, the model performs really well in terms of providing latent dimension low or high because it still provides good RMSE values near 0.

We can also check the axes values getting interchanged and then the corresponding values at the final axis to be interpreted. As we have latent dimension 32 to provide better results. We will take this as a reference value and then analyze the axes changes. As in figure 6.14 (1), we could see the model performance being the lowest in RMSE error values of all the remaining hyperparameter values. Figure 6.14 (2) shows the learning rate hyperparameter placed near the final result axes, and we can interpret the learning rate value, which provides a low error rate (RMSE). As we can see from the graph, from the minimum to the maximum, the RMSE error value also goes from the minimum to the maximum. So, the learning rate performs well at lower values in the order of 0.0001, 0.001, 0.0020, 0.0040, and 0.0080.

Figure 6.14 (3) shows the regularization value placed before the RMSE axis, for which the values can be read accordingly. Here, we can see from the figure that values 0.0010 and 0.010 produce better results compared to the remaining values. Even though we have a lower value of 0.0001 than those two values, here the interpretation won't work like lower the value, better the performance according to the results. Figure 6.14 (4) depicts the dropout value placed before the last axis, for

which the values can be read accordingly. From this visualization, we can convey from the provided dropout values of 0.1, 0.01, 0.001, 0.02, 0.002 that the lowest RMSE value producing is 0.001, followed by 0.002. From this, we could draw the conclusion that even with dropout, the lower the value, the better the performance of the model.

We can check the same for the remaining latent dimension values as in figure 6.15, where (5), (6), (7), and (8) represent the latent dimensions of 128, 64, 16, and 8, respectively. From the figures 6.15 (5) and (6), we can see that the RMSE value increases beyond the normal range of other latent dimensions. So, clearly, we can avoid considering the latent dimensions 128 and 64 in the NCF model. Figures 6.15 (7) and (8) depict the parallel coordinate plots for the corresponding latent dimensions. Even though it provides decent RMSE values with the combination of different hyperparameters, it is not performing better than the latent dimension with a 32-dimensional value. We can also analyze the parallel plot in a similar way by changing the axis before the RMSE axis and learning the performance of the hyperparameters.

A parallel coordinate plot is really helpful in understanding the models' performance with the hyperparameter range of selection. From the analysis, we can conclude that the model performs better with the latent dimension value 32 than with the other latent dimension values. This helps to finalize how we can proceed further to build the model, with the latent dimension value fixed at 32 and the choice of remaining hyperparameter values that have a hit on the RMSE value that makes the model's performance degrade.

6.2.5 Analysis 5: Combined view of the NCF, GMF and MLP model

Neural collaborative filtering (NCF) method is an architecture of ensemble methods like combination of General Matrix Factorization (GMF) and MultiLayer Perceptron (MLP). So, in order to understand the models' pattern, we combine the output values from each model, and then, by applying dimensionality reduction techniques, we reduce the values in 2 dimensions and then plot the confined points as a scatter plot to find the relationship. The combined view is displayed in the figure 5.5 (9), which is the view with all the outputs from the models represented in a single 2D space. It has a filter for selecting particular methods, and we can see how the respective model is represented with the color scale of the remaining methods partially highlighted in the background. This really helps to understand the models' pattern in regards to the remaining model in the background. Figure 6.16, shows the GMF output with red color data points highlighted and the remaining model NCF and MLP points dimmed. From this figure, we can identify the strength of the GMF in certain areas and where certain points are correlated with the other model points. We could clearly see the red color to be more distributed between the range of (-30, -70) of the x-axis and (10, 40) of the y-axis (the zoomed part to the left of the main visualization figure 6.16), which simplifies the fact that the model performance towards this area is only with the GMF outputs and there is no influence from other methods. We could see clearly that the remaining points are more correlated either with MLP, with NCF, or with both NCF and MLP. We can also drag with the select option that filters points, and the corresponding points will be displayed in the below table with the respective t-SNE values along with the customer id and the corresponding source of points, where 1, 2, and 3 represent the GMF, MLP, and NCF, respectively.

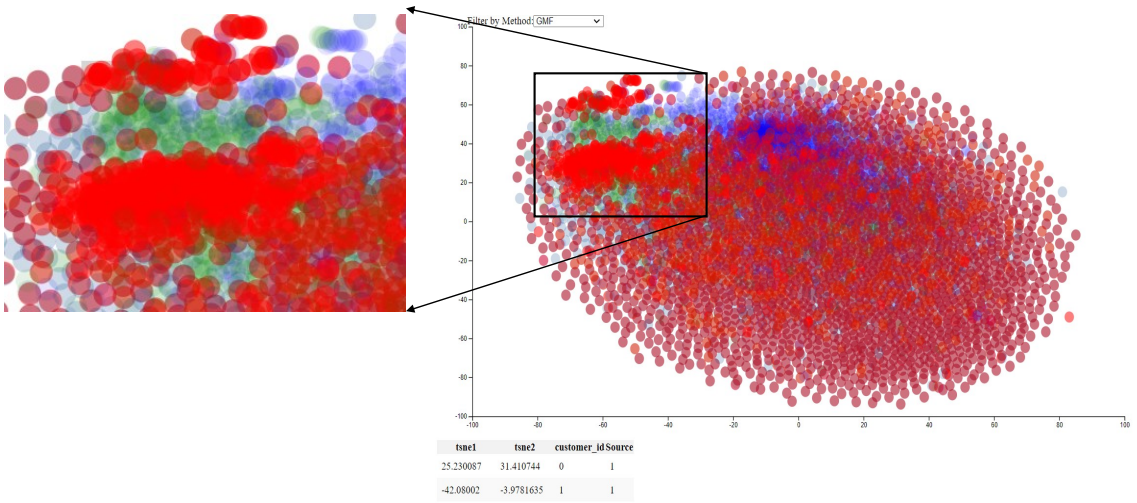

Figure 6.16: Results analysis with dimensionality reduction technique tSNE to be in one combined view of MLP, GMF, and NeuMF combination of NMF and GMF, highlighting GMF and diminishing remaining NCF and GMF output. Colorcodings are represented by ● as GMF, ● as MLP, and ● as NCF.

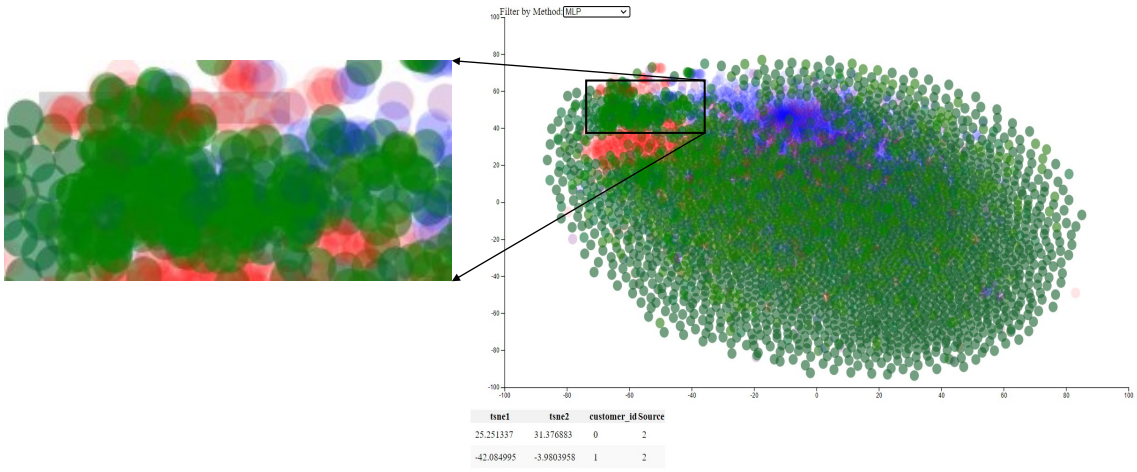

Figure 6.17: Results analysis with dimensionality reduction technique tSNE to be in one combined view of MLP, GMF, and NeuMF combination of NMF and GMF, highlighting MLP and diminishing remaining NCF and GMF output. Colorcodings are represented by ● as GMF, ● as MLP, and ● as NCF.

Figure 6.17, visualizes the MLP output with green color points highlighted and the remaining model GMF and NCF points dimmed. From this figure, we can identify the strength of the MLP in certain areas and where certain points are correlated with the other model points. We could clearly see the green color to be more distributed between the range of (-50, -70) of the x-axis and (40, 60) of the y-axis (the zoomed part to the left of the main visualization figure 6.17), which simplifies the fact that the model performance towards this area is only with the MLP outputs and there is no influence from other methods. We could see clearly that the remaining points are more correlated either with the GMF, the NCF, or both the GMF and the MLP.

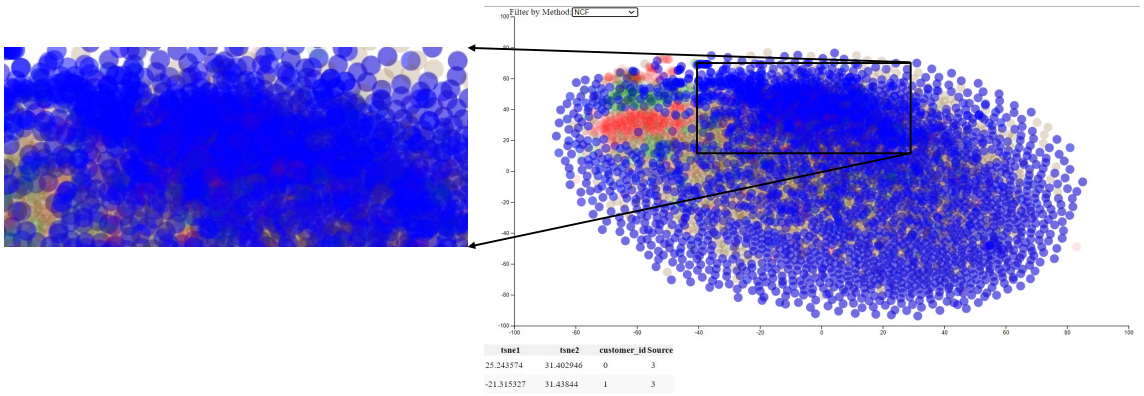

Figure 6.18: Results analysis with dimensionality reduction technique tSNE to be in one combined view of MLP, GMF, and NeuMF combination of NMF and GMF, highlighting NCF and diminishing remaining MLP and GMF output. Colorcodings are represented by ● as GMF, ● as MLP, and ● as NCF.

In the figure 6.18, we could see the pattern is more from the NCF influence because the majority of the figure has high intensity of the color blue in the center area, i.e., (-30, 40) in the x-axis and (0, 60) in the y-axis (the zoomed part to the left of the main visualization figure 6.18). But there are some points in that area that were from either MLP or GMF or both values that were highly correlated. The points on the edge are mostly mixed with red, which signifies the GMF output. So, the NCF model is learned as a combination of GMF and MLP, and it also learns from both models to provide value.

From all these figures and interpretations, we can assure ourselves that NCF learning is a combination of MLP and GMF. In which the linear features are learned from GMF as it involves the matrix multiplication of users and items, and the MLP to learn the non-linear features from the provided inputs with the advent of the hidden layers. This analysis clearly supports the NCF as an ensemble method to learn the inputs from both the GMF and MLP models and then represent them as closely correlated.

6.2.6 Analysis 6: Visualization of hidden layers of MLP and concatenation layer

Through this analysis,the output of each hidden layer from the multilayer perceptron (MLP) is derived along with the concatenation layer of Neural Collaborative Filtering (NCF) model. The hidden layers in MLP are 64, 32, 16, and 8, and then the final concatenation layer is 1. So, for determining which feature is important and identifying which neurons are more active or responsive to specific features or patterns in the input data, we derive the activation functions from each hidden layer and plot them as a heatmap. Additionally, we can select a particular point, and its corresponding value will be displayed in the bottom box view. This information can help understand how the neural network is encoding and processing information at a particular dense layer. For instance, the figure 6.19 (a) is the output representation of 64 inputs and their corresponding 64 output values. From this heatmap, we can generate insights into how the neurons are contributing to the output representation. Based on the values, the color scale gets varied for each value. The color intensity is represented by the color scale of red, with high and low values. Similarly, the figures

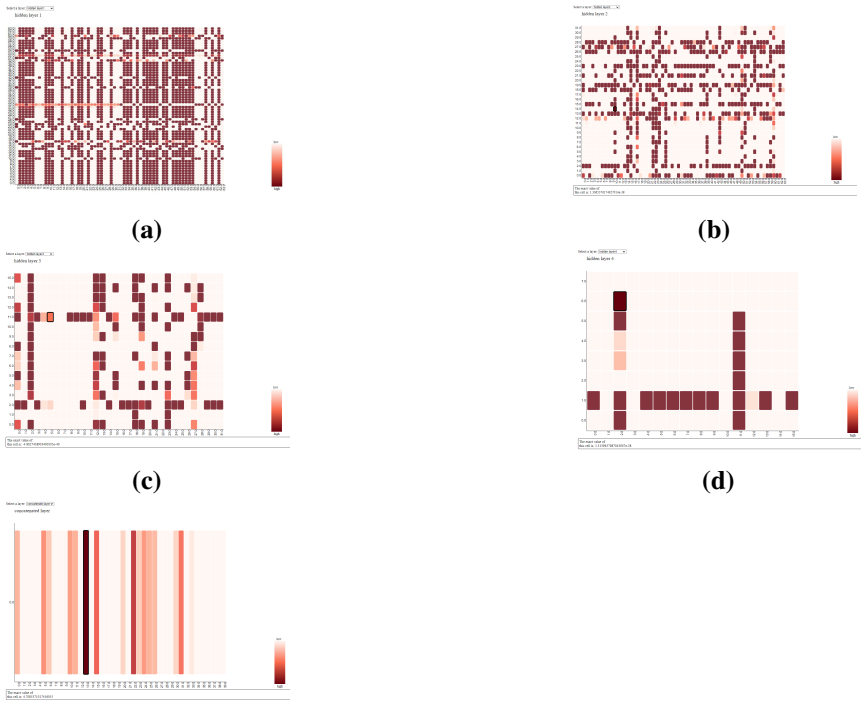

Figure 6.19: Plots of hidden layers derived from Neural Collaborative Filtering's multilayer perceptron (MLP) part and finally display the concatenated layer output from both MLP and General Matrix Factorization (GMF) layers' output, with x-axis being input layer values and y-axis beng output layer values. (a): Hidden layer 1 of MLP, (b): Hidden layer 2 of MLP, (c): Hidden layer 3 of MLP, (d): Hidden layer 4 of MLP, (e): Concatenated layer of NCF.

6.19 (b), (c), and (d) represent the heatmap of hidden layer 2 with 64 as input and 32 as output, which is of 64×32 shape; the heatmap of hidden layer 3 with 32 as input and 16 as output, which is of 32×16 shape; and the heatmap of hidden layer 4 with 16 as input and 8 as output, which is of 16×8 shape, respectively. The final concatenation layer is the heatmap representation of the concatenation of MLP and General Matrix Factorization (GMF) inputs and the corresponding output 1, which is 16×1 shape. As for the insights derived we could confirm about the feature importance of each neuron with respective to each input and output dimensions respectively.

6.2.7 Analysis 7: Learning from matrix bandwidth minimization

If we represent the large sparse matrix in a heatmap form, the values will be more dispersed, and we cannot get any insights from the visualization. And also, to handle the large sparse matrix, we need to reduce the shape or find a way to reduce the matrix. For this purpose, we use the compressed sparse row (CSR) or compressed row storage (CRS) format, which represents a sparse matrix M by three (one-dimensional) arrays, which are row, column, and the corresponding non-sparse value. This saves the representation of sparse matrix, which have a large percentage of zero values, which in turn saves the memory required to save the matrix and the computation required for handling

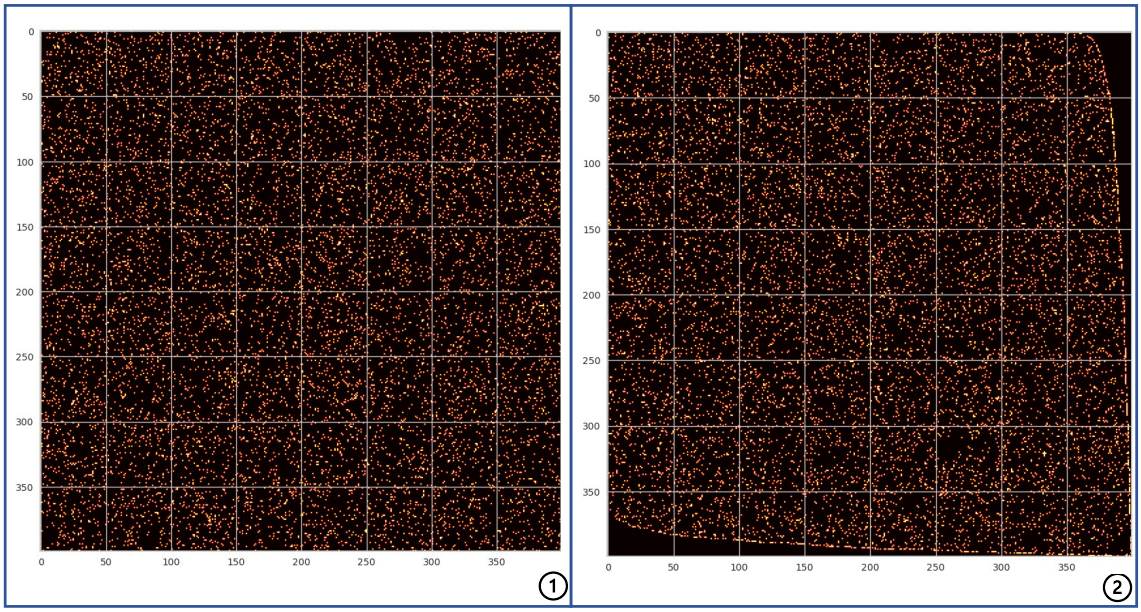

Figure 6.20: Represented the output matrix of the NCF model as a 400 × 400 matrix with the help of the Reverse Cuthill-Mckee method. (1).represents the matrix without any method applied; (2). represents the matrix with Reverse Cuthill-Mckee applied.

the sparse matrix. In our case of a sparse matrix, the percentage of zeros present is 93.5%, which consumes a lot of space if stored directly. So, by applying the CSR format, the memory required to save the matrix is drastically reduced from 75.7 MB to 0.06 MB.

After reduction, we will have values greater than 0. To find the pattern that it can generate and derive insights from it, we tried matrix bandwidth minimization techniques. Matrix Bandwidth Minimization is the method used to reduce the bandwidth of the provided matrix. The bandwidth of the matrix refers to the measure of how far the matrix elements are from the main diagonal of the matrix. It is used to reduce the distance between any non-zero element and the main diagonal. One of the algorithms that do this function is the Reverse Cuthill-Mckee method, where the method starts with a random node of a row or column in the matrix. Then it performs a breadth-first search (BFS) from that selected node, which visits its neighbors and labels other nodes' distance from that starting node, and then it sorts the nodes in ascending order based on the level. This will be an iterative approach until it visits all the nodes that are present in the matrix. Later, the nodes are ordered in reverse, and according to the reversed order, the rows and columns of the matrix are reordered. Reorder the rows and columns of the matrix according to the reversed order.

Figure 6.20 represents the output matrix of the NCF model reduced to a 400 × 400 matrix. Figure 6.20 (1) is the generic output without any methods applied. Figure 6.20 (2) is the matrix output with the reverse Cuthill-Mckee method. From this figure, we can see that the reverse Cuthill-McKee did not work to bring the matrix values to the center of the diagonal, which fails the necessary requirement of the matrix bandwidth minimization method. In Figure 6.20 (2), we can see that the clusters are formed in the graph as an artificial layer without much insight from the visualization. So, it's evident that even with matrix bandwidth minimization, the evaluation of large sparse matrices is complex, and in this case, there won't be any value created from them.

7 Conclusion

In this chapter, we will discuss the findings that we discovered by using the visualization techniques applied to the product bundling recommendation model with machine and deep learning approaches and insights about the product bundling recommendation model.

7.1 Summary

We have implemented the product bundling algorithm using Neural Collaborative Filtering (NCF), Alternate Least Square (ALS), and frequent itemset algorithms for the purpose of providing the product bundles to the customer with personalization and to increase cross-selling for the organization. Based on the problem statement, we have built the visualization tool, which helps in understanding the internal workings of NCF and getting insights about the results that are derived. The pilot interview was conducted as per the visualization study and gathered requirements from both technical and non-technical people that helped in building the visualization tool. With the help of the visualization tool, we explored the NCF models' workings by using dimensionality reduction techniques and plotting them. This helped to understand how the pattern of NCF is getting influenced by the two other models, namely General Matrix Factorization (GMF) and MultiLayer Perceptron (MLP). These two models' patterns are analyzed and then correlated with the pattern of the NCF model. We could see that the patterns that are in the GMF and MLP will be in combination or contribute separately to the results in the NCF pattern. We have analyzed the scatter plot in different views and in a combined view of all the models' results. This helped in understanding the customers who got recommended the new product based on the customers in the same cluster group as that particular customer. We have successfully explored the internal states of the NCF model and interpreted the models' recommendation results, which helped to understand the usage of the NCF model as the recommendation engine, which takes care of the customers' personalization requirements. We have also used heatmap output to understand which neurons contribute more to the output in the hidden layers, the output representation of each hidden layer, and finally, the output layer. The hyperparameter parallel coordinate plot is used to find the hyperparameter configuration that leads to better model performance, which helped in understanding how the model performs well with different hyperparameter configurations. These results are similar to hyperparameter tuning but provide an overall view of how the model performs compared to other hyperparameter values. Another requirement of the visualization tool is to understand the existing product relationships and provide a KPI-based view. For this purpose, we have used a network graph with the help of results from frequent itemset mining, which was used to find the relationship between the products that are often bought together. This provides better insights into the existing relationships with the products by analyzing the pattern of frequently purchased items together. With the help of a network graph, we identified the relationships between the products, matched those results to individual customers, and then provided them with the bundle. For the KPI requirement of displaying the

sales value in each hierarchical view, we have used hierarchical edge bundling to reduce the clutter caused by a standard table-based view to understand the sales value generated in each hierarchical level. This served the purpose of understanding the hierarchy level of products, and this gave better insights on which product level the management should keep track of to understand the grain level details of the product hierarchy. With the expert study conducted to get the end user's perspective, the purpose of the visualization toolbox is evaluated with the expert study for providing the requirements stated in the pilot interview and ensured reusability of the tool for the future purposes. As with the recommendation model evaluation with expert study, the currently built models comprise cross-selling recommendations considering the customers' personalization factor, as clearly depicted with the backing visualizations of pattern analysis with the NCF model and network graph analysis of frequent itemset mining results. As a result, the goal of interpreting the NCF model and how the recommendations are generated with the help of a visualization tool and recommending the product bundles with customer personalization and cross-selling is achieved.

7.2 Future Scopes

With the developed method, we can further include the following points in the direction of the future scope to be explored:

- In the current NCF model, we have only considered customer and product features. But for further optimization, we can include the region values, customer divisions, and specific sales parameters that will be useful for the model to provide more personalized recommendations to the customers, as these features add more importance to being learned by the model and providing recommendations.
- The current analysis is done with the final layers of each model, and then the models' interpretability is handled, but to understand it much more profound, we could study more about the hidden layers of MLP and then compare the results from it to check the models' internal layers and their results from the initial input to the final output. If further research is made in this direction, it would be better to understand the non-linear learning in the MLP model.
- Since ensemble learning is a combination of GMF and MLP, the applicability of LIME and SHAP model explainability methods failed with the gradient backtracking that used to happen generally with the model explainability methods. With the NCF, the backtracking issue arises because the model's architecture lacks a clear and direct connection between input features and predictions. The non-linear transformations and embedding layers make it challenging to trace how changes in individual input features propagate through the model. Consequently, the gradients become difficult to compute accurately or are not readily available for specific model components. So, non-linear models should be explored to interpret the results.
- The product bundling results are based on the products that are often bought together and the customers' purchase history. But the product bundling must be analyzed with the help of products used together, which was the limitation after the evaluation results. The data in the current infrastructure did not capture the details of the products used together. So, in the future, that data can be used on top of the results obtained with the current approach, which might provide products that always go together but not with cross-selling feasibility.

- Currently, the NCF model is evaluated with only four hyperparameters; batch size and epoch could be used to further optimize the model's performance. Here, we have fixed 128 batch sizes and 30 epochs with the hyperparameter default tuning results. If we had investigated the models' performance with different batch size values and epochs, it would have provided more performance insights.
- K-means clustering applied to the pattern analysis of the NCF, GMF, and MLP models did not provide any insights from the results or any proper cluster groups. This provides more room to analyze further clustering algorithms like spectral clustering, fuzzy clustering, and DBSCAN clustering. These algorithms might produce clusters that provide more insights.

Bibliography

[AIS93] R. Agrawal, T. Imieliński, A. Swami. “Mining Association Rules between Sets of Items in Large Databases”. In: *SIGMOD Rec.* 22.2 (June 1993), pp. 207–216. ISSN: 0163-5808. DOI: 10.1145/170036.170072 (cit. on p. 41).

[AN17] B. Abdollahi, O. Nasraoui. “Using Explainability for Constrained Matrix Factorization”. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys ’17. Como, Italy: Association for Computing Machinery, 2017, pp. 79–83. ISBN: 9781450346528. DOI: 10.1145/3109859.3109913 (cit. on p. 34).

[BGB15] P. B.Thorat, R. Goudar, S. Barve. “Survey on Collaborative Filtering, Content-Based Filtering and Hybrid Recommendation System”. In: *International Journal of Computer Applications* 110 (Jan. 2015), pp. 31–36. DOI: 10.5120/19308-0760 (cit. on p. 27).

[BGM15] D. Bokde, S. Girase, D. Mukhopadhyay. “Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey”. In: vol. 49. Apr. 2015. DOI: 10.1016/j.procs.2015.04.237 (cit. on p. 31).

[BLT17] K. Bauman, B. Liu, A. Tuzhilin. “Aspect Based Recommendations: Recommending Items with the Most Valuable Aspects Based on User Reviews”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’17. Halifax, NS, Canada: Association for Computing Machinery, 2017, pp. 717–725. ISBN: 9781450348874. DOI: 10.1145/3097983.3098170 (cit. on p. 34).

[BOH11] M. Bostock, V. Ogievetsky, J. Heer. “D3: Data-Driven Documents”. In: *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011). DOI: 10.1109/TVCG.2011.185 (cit. on p. 61).

[BR10] M. Bharati, B. Ramageri. “Data Mining Techniques and Applications”. In: *Indian Journal of Computer Science and Engineering* 1 (Dec. 2010) (cit. on p. 24).

[Cac11] Cacheda, Fidel and Carneiro, Vıéctor and Fernández, Diego and Formoso, Vreixo. “Comparison of Collaborative Filtering Algorithms: Limitations of Current Techniques and Proposals for Scalable, High-Performance Recommender Systems”. In: *ACM Trans. Web* 5.1 (Feb. 2011). ISSN: 1559-1131. DOI: 10.1145/1921591.1921593 (cit. on p. 28).

[CFH12] S. Cleger-Tamayo, J. M. Fernandez-Luna, J. F. Huete. “Explaining Neighborhood-Based Recommendations”. In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’12. Portland, Oregon, USA: Association for Computing Machinery, 2012, pp. 1063–1064. ISBN: 9781450314725. DOI: 10.1145/2348283.2348470 (cit. on p. 33).

[DWHDL20] Databricks. *What Is a Lakehouse?* 2020. URL: https://www.databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html (cit. on p. 38).

[FPS96] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth. "The KDD Process for Extracting Useful Knowledge from Volumes of Data". In: *Commun. ACM* 39.11 (Nov. 1996), pp. 27–34. ISSN: 0001-0782. DOI: 10.1145/240455.240464 (cit. on pp. 24, 26).

[GH16] C. A. Gomez-Uribe, N. Hunt. "The Netflix Recommender System: Algorithms, Business Value, and Innovation". In: *ACM Trans. Manage. Inf. Syst.* 6.4 (Dec. 2016). ISSN: 2158-656X. DOI: 10.1145/2843948 (cit. on p. 27).

[GNOT92] D. Goldberg, D. Nichols, B. M. Oki, D. Terry. "Using Collaborative Filtering to Weave an Information Tapestry". In: *Commun. ACM* 35.12 (Dec. 1992), pp. 61–70. ISSN: 0001-0782. DOI: 10.1145/138859.138867 (cit. on p. 31).

[HKR02] J. Herlocker, J. A. Konstan, J. Riedl. "An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms". In: *Information Retrieval* 5.4 (Oct. 1, 2002), pp. 287–310. ISSN: 1573-7659. DOI: 10.1023/A:1020443909834. (Visited on 05/14/2023) (cit. on p. 31).

[HLZ+17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua. "Neural Collaborative Filtering". In: *CoRR* abs/1708.05031 (2017). DOI: 10.1145/3038912.3052569. arXiv: 1708.05031 (cit. on pp. 32, 46, 58).

[IFO15] F. Isinkaye, Y. Folajimi, B. Ojokoh. "Recommendation Systems: Principles, Methods and Evaluation". In: *Egyptian Informatics Journal* 16.3 (2015), pp. 261–273. ISSN: 1110-8665. DOI: 10.1016/j.eij.2015.06.005 (cit. on pp. 28, 29).

[JIA21] L. "JIANGSU ZHONGTIAN ANCHI TECHNOLOGY CO. *"The Large-Scale Application of Vision Algorithm Technology in the Field of Commercial Vehicle Security Operation and Maintenance"*. "[Online; accessed 08-April-2023]". 2021 (cit. on p. 23).

[Jör17] Jöran Beel. "Towards Effective Research-Paper Recommender Systems and User Modeling based on Mind Maps". In: *CoRR* abs/1703.09109 (2017). DOI: 10.25673/4222 (cit. on p. 31).

[JSK10] G. Jawaheer, M. Szomszor, P. Kostkova. "Comparison of Implicit and Explicit Feedback from an Online Music Recommendation Service". In: *HetRec '10*. 2010. DOI: 10.1145/1869446.1869453 (cit. on p. 31).

[KAU16] S. Khusro, Z. Ali, I. Ullah. "Recommender Systems: Issues, Challenges, and Research Opportunities". In: Feb. 2016, pp. 1179–1189. ISBN: 978-981-10-0556-5. DOI: 10.1007/978-981-10-0557-2_112 (cit. on p. 29).

[KLM96] L. P. Kaelbling, M. L. Littman, A. W. Moore. "Reinforcement Learning: A Survey". In: *CoRR* cs.AI/9605103 (1996). DOI: 10.1613/jair.301 (cit. on p. 22).

[LKF15] S. Li, J. Kawale, Y. Fu. "Deep Collaborative Filtering via Marginalized Denoising Auto-encoder". In: Oct. 2015, pp. 811–820. DOI: 10.1145/2806416.2806527 (cit. on p. 32).

[LL17] S. M. Lundberg, S. Lee. "A Unified Approach to Interpreting Model Predictions". In: *CoRR* abs/1705.07874 (2017). DOI: 10.48550/arXiv.1705.07874 (cit. on p. 35).

[LRC+18] Y. Lin, P. Ren, Z. Chen, Z. Ren, J. Ma, M. de Rijke. “Explainable Fashion Recommendation with Joint Outfit Matching and Comment Generation”. In: *CoRR* abs/1806.08977 (2018). DOI: 10.1109/TKDE.2019.2906190 (cit. on p. 34).

[MD20] M. Meyer, J. Dykes. “Criteria for Rigor in Visualization Design Study”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 87–97. DOI: 10.1109/TVCG.2019.2934539 (cit. on pp. 17, 33, 64, 65).

[PJK+17] H. Park, H. Jeon, J. Kim, B. Ahn, U. Kang. “UniWalk: Explainable and Accurate Recommendation for Rating and Network Data”. In: *CoRR* abs/1710.07134 (2017). DOI: 10.1145/3308558.3313607 (cit. on p. 34).

[PW18] G. Peake, J. Wang. “Explanation Mining: Post Hoc Interpretability of Latent Factor Models for Recommendation Systems”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & amp; Data Mining*. KDD ’18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 2060–2069. ISBN: 9781450355520. DOI: 10.1145/3219819.3220072 (cit. on p. 35).

[RIS+94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl. “GroupLens: An Open Architecture for Collaborative Filtering of Netnews”. In: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. CSCW ’94. Chapel Hill, North Carolina, USA: Association for Computing Machinery, 1994, pp. 175–186. ISBN: 0897916891. DOI: 10.1145/192844.192905 (cit. on p. 33).

[RLL+17] Z. Ren, S. Liang, P. Li, S. Wang, M. de Rijke. “Social Collaborative Viewpoint Regression with Explainable Recommendations”. In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. WSDM ’17. Cambridge, United Kingdom: Association for Computing Machinery, 2017, pp. 485–494. ISBN: 9781450346757. DOI: 10.1145/3018661.3018686 (cit. on p. 34).

[RRS11] F. Ricci, L. Rokach, B. Shapira. “Introduction to Recommender Systems Handbook”. In: *Recommender Systems Handbook*. Ed. by F. Ricci, L. Rokach, B. Shapira, P. B. Kantor. Boston, MA: Springer US, 2011, pp. 1–35. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_1 (cit. on pp. 33, 34).

[RSG16] M. T. Ribeiro, S. Singh, C. Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *CoRR* abs/1602.04938 (2016). DOI: 10.48550/arXiv.1602.04938 (cit. on p. 35).

[Sar21] I. Sarker. “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions”. In: *SN Computer Science* 8 (Feb. 2021). DOI: 10.1007/s42979-021-00815-1 (cit. on p. 22).

[SB17] S. Sharma, S. Bhatia. “A Study of Frequent Itemset Mining Techniques”. In: *International Journal of Engineering & Technology* 6 (Oct. 2017), p. 141. DOI: 10.14419/ijet.v6i4.8300 (cit. on pp. 15, 32).

[SC12] R. Singh, S. Chaudhary. “Data Mining Approach Using Apriori Algorithm: The Review”. In: *IOSR Journal of Electronics and Communication Engineering* 4 (2012), pp. 12–15. DOI: 10.9790/2834-0421215 (cit. on p. 43).

[Sch07] Schafer, Ben and Frankowski, Dan and Dan, and Herlocker, and Jon, and Shilad, and Sen, Shilad. “Collaborative Filtering Recommender Systems”. In: Jan. 2007. DOI: 10.1007/978-3-540-72079-9_9 (cit. on p. 15).

[SKKR01] B. Sarwar, G. Karypis, J. Konstan, J. Riedl. “Item-Based Collaborative Filtering Recommendation Algorithms”. In: *Proceedings of the 10th International Conference on World Wide Web*. WWW ’01. Hong Kong, Hong Kong: Association for Computing Machinery, 2001, pp. 285–295. ISBN: 1581133480. DOI: 10.1145/371920.372071 (cit. on p. 32).

[SMH07] R. Salakhutdinov, A. Mnih, G. Hinton. “Restricted Boltzmann Machines for Collaborative Filtering”. In: vol. 227. June 2007, pp. 791–798. DOI: 10.1145/1273496.1273596 (cit. on p. 32).

[SMM12] M. Sedlmair, M. Meyer, T. Munzner. “Design Study Methodology: Reflections from the Trenches and the Stacks”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2431–2440. DOI: 10.1109/TVCG.2012.213 (cit. on pp. 17, 18, 33, 63).

[SMSX15] S. Sedhain, A. Menon, S. Sanner, L. Xie. “AutoRec: Autoencoders Meet Collaborative Filtering”. In: May 2015, pp. 111–112. DOI: 10.1145/2740908.2742726 (cit. on p. 32).

[TJWW19] Y. Tao, Y. Jia, N. Wang, H. Wang. “The FacT: Taming Latent Factor Models for Explainability with Factorization Trees”. In: *CoRR* abs/1906.02037 (2019). DOI: 10.1145/3331184.3331244 (cit. on p. 34).

[TT12] G. Takács, D. Tikk. “Alternating Least Squares for Personalized Ranking”. In: (Sept. 2012). DOI: 10.1145/2365952.2365972 (cit. on pp. 15, 32, 43).

[TUR50] TURING, A. M. “I.—Computing Machinery And Intelligence”. In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433 (cit. on p. 21).

[VSR09] J. Vig, S. Sen, J. Riedl. “Tagsplanations: Explaining Recommendations Using Tags”. In: *Proceedings of the 14th International Conference on Intelligent User Interfaces*. IUI ’09. Sanibel Island, Florida, USA: Association for Computing Machinery, 2009, pp. 47–56. ISBN: 9781605581682. DOI: 10.1145/1502650.1502661 (cit. on p. 34).

[WCY+18] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, X. Xie. “A Reinforcement Learning Framework for Explainable Recommendation”. In: *2018 IEEE International Conference on Data Mining (ICDM)*. 2018, pp. 587–596. DOI: 10.1109/ICDM.2018.00074 (cit. on p. 33).

[ZC18] Y. Zhang, X. Chen. “Explainable Recommendation: A Survey and New Perspectives”. In: *CoRR* abs/1804.11192 (2018). DOI: 10.1561/1500000066 (cit. on p. 33).

[ZLZ+14] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, S. Ma. “Explicit Factor Models for Explainable Recommendation Based on Phrase-Level Sentiment Analysis”. In: *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR ’14. Gold Coast, Queensland, Australia: Association for Computing Machinery, 2014, pp. 83–92. ISBN: 9781450322577. DOI: 10.1145/2600428.2609579 (cit. on pp. 33, 34).

All links were last followed on July 14, 2023.

A Appendix

In this section, I will present a thorough overview of my work conducted during my master's thesis while working at Wuerth. To ensure transparency, I will provide a detailed account of the entire process, starting from the initial stages and concluding with the final outcomes. This will include an examination of the challenges I encountered, the results I achieved, and the overall workflow I implemented throughout the thesis.

My role at Würth was to build a product bundling recommendation algorithm by researching the applicability of machine or deep learning methods. I worked directly with the VSSB-Big Data Analytics team in the VSS (Strategic Sales Operations) department. The Strategic Sales Operations department is responsible for controlling and driving the sales of AWKG (Adolf Würth KG). The VSSB big data team is responsible for building data science solutions that drive the sales value of AWKG. The main goal of my work is to build the product bundling recommendation according to each customer's personalized choices, and with this, gross merchandise volume and sales can be leveraged by taking care of customer satisfaction. In the first week, I was introduced to all the teams in the VSS department. There were six teams under VSS, including VSSB. The teams are VSSP (Pricing team), VSSM (Motivation & Benefits), VSSA (Structure & Analysis), VSSD (Projects & Business Development), VSSE (Customer Experience Management), and VSSB (Big Data Analytics). This gave a clear overview of how the business process is done across the team and how they organize to work as a single department to motivate the sales representative for high performance, healthy growth in terms of sales and customers, steering business with the help of data-driven sales, the possibilities of omni-channel sales, and analysis on top of it. Then, for the process of knowing the data infrastructure, I worked with the BDO (big data Operation). BDO is responsible for collecting data, integrating it, and then preparing the data set that is made available on the data lake. Datalake consists of unstructured, structured, and semi-structured data. I self-organized myself to setup the tools and accesses that are required for starting up the thesis project.

The timeline of the project is decided with the help of the CRISP-DM method, which involved business, data understanding, dataset preparation, benchmarking the results, results evaluation, a visualization pilot interview, developing a visualization tool, and evaluating it. A phase of two weeks was spent introducing the team, the business process, the current projects that are done by all teams, and, more specifically, the projects done by the VSSB department. I started to work with five data scientists and two managers on the VSSB team. I have conducted meetings with a person from all the teams to understand the real need for product bundling and started to correlate things that are required to build the Machine learning model and drive the literature review accordingly. Before starting with my thesis, there was no recommender system model already made by the team. So, we have decided to build a minimum viable product (MVP) that can be modeled, evaluated, and deployed within a short span of time. My supervisor for this project is Akhileshwar Sharma, a senior data scientist. With whom I had regular meetings once a week to share the insights and work results that I got in the week. This helped align with the business requirements and provided

technical feasibility for proceeding with the model building, evaluation, and feedback. Then, later, for six weeks, I started with the literature review for the product bundling and recommendation systems. Meanwhile, understanding the product ecosystem, sales process, and dataset identification Based on these steps, we finalized the datasets that are needed for the project and started to prepare the dataset by doing exploratory data analysis. At this stage, there were a lot of dataset issues because of the lack of standardized data. There were changes happening to the referred data often. At first, I started with the dataset that was smaller than the actual data. Later, after addressing this issue by comparing the source data, the dataset was corrected for the integration issues. There were data quality issues, which were also reported and rectified along with the BDO IT team during the phase of dataset preparation for the model. After these steps, finalize the methods and approach that will be used to proceed with the product bundling recommendation. This is finalized after presenting it to the team along with the managers to get their approval for starting with the presented approach. The methods that are accepted are Neural collaborative filtering, alternate least squares and frequent itemset mining. After selecting the dataset, we did feature engineering to understand which features would be helpful for the model. And built a pipeline for creating the dataset that will be needed for the model. The model building phase took over six weeks to develop all three models, which also included building product bundles out of all these approaches. And finally, I presented each model's results and the evaluation that was carried out to assess the model's performance and benchmark it. During this phase, I included other team members to get their different opinions and insights to improve the existing approach.

Later, the management did not agree with the NCF and ALS results as they didn't have a proper, backed-up reason for the recommendation results that were produced, which they thought might cause dissatisfaction from the customer's point of view. As a result of this decision, frequent itemset mining is suggested, and to handle edge cases like the cold start problem, we employed item-item collaborative filtering, to which they agreed since the products that were suggested to the customers were taken from the rule and score that were generated by both models. In the cases of NCF and ALS, they provided new products to the customer but lacked explanation or interpretability. This clearly paved the way to build the visualization tool with the possibility of model interpretability or post hoc explanation. This process took around four weeks to get the opinion of the management on the results.

In the next four-six weeks, I started to evaluate the model by closely working with the international sales team for the construction customer division. There, I provided the team with results that are easily understandable with the Power BI dashboard by developing the tables and some KPIs' of the frequent itemset mining and item-item collaborative filtering methods. This process took a long time because the team needed time to evaluate and because of the unavailability of team members. But with this evaluation, there was no progress since they gave feedback that the process could only be tested live and they couldn't provide any insights. Later, I proceeded with the national sales team for the metal customer division. Since I have developed all the models and results for all the customer divisions, it was easier to follow up with the team and provide them with the results soon. With this team also, I prepared a dashboard and then presented it to them. It took two weeks to schedule a meeting with the sales representative and product owners to analyze the results in detail. They came with the suggestion that this result can be used for the purpose of cross-selling as the customers will benefit from the recommendation, but not as a product bundle since the customers are craftsmen and they need products that are used together, not as a cross-selling suggestion or bought together. After these results, the project was reiterated to find the usage of the products together rather than being bought together, which was the result that was presented. But those data

were not captured in the existing data set. The findings from this thesis are that we needed to include the customer in the initial phase of testing and requirement gathering and that the requirements should be mended properly before starting the project. Since the project's main goal was to provide cross-selling, it later evolved that this was not the case for product bundling from the customer's point of view. This involved working with different team members who have been in continuous contact with the customer. I proposed a few changes to the management with these findings, and they are in the process of changing the model-building approach generally. I have prepared the model, documented the findings, and provided the code and method results, so one can easily start further building and deploying it in the future.

In the mid-phase, after presenting the results to the management team, they required explanations for the recommender systems built using NCF. I started to conduct the pilot interview with the data scientist and manager from the VSSB team with regard to building the visualization tool on the basis of the recommender system and further requests that can be leveraged with the help of visualization. The visualization tool's building took nearly three months. I have constant communication regarding the tool and its development with both supervisors from the company and the university. From these inputs, I have built, improved, and presented the tool to the people involved in the pilot study. And conducted expert study of the tool to determine whether it covered the aspects that they had requested, and I have also documented those results for future usage if anyone was required to use the tool. Apart from this, we also had brainstorming sessions to use my previous experience with the existing projects to develop them further.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature