Recommendation Engines in Private Banks: Personalized Asset Management

Arian Madadi
madadi.arian@u.nus.edu

***Abstract*—Recommendation systems are increasingly shaping industries, driving user engagement and business growth. For the banking sector, and specifically for Julius Baer, existing systems excel at initial recommendations, but lack in customer personalization and adaptability. This paper presents a versatile recommendation system for Julius Baer, employing LGBMRank and retraining systems to improve personalization, as well as a dashboard for data, performance, and recommendation visualization. This approach integrates traditional recommendation methods with a monitoring system that tracks the model's performance and retrains it when a drop in performance is detected. This enables Julius Baer to consistently provide recommendations that are both relevant and responsive to evolving market conditions and client needs. It not only heightens client satisfaction and engagement, but also optimizes resource use by eliminating the need for constant retraining.**

***Index Terms*—Recommendation System, User-user Collaborative Filtering, Julius Baer Singapore, Performance Monitoring.**

I. Introduction

A. Business Climate and Motivation

The digital age has ushered in an era of abundant choices for consumers, making it difficult for businesses to offer generic services. Recommendation systems have been adopted as a primary solution for this complexity, allowing them to cater more to individual user preferences. By personalizing user experiences, these systems not only enhance user engagement and retention, but also help foster brand loyalty and trust [1].

Prominent platforms like YouTube and Netflix truly display the power of recommendation systems. As of 2018, YouTube's recommendation algorithm was responsible for suggesting 70 percent of the content viewed by its users [2]. Similarly, in 2017, Netflix reported that 80 percent of its viewed content stemmed from its recommendation system, rather than direct searches [3]. Beyond streaming platforms, recommendation systems have begun to permeate in diverse sectors, from e-commerce companies such as Amazon, to social media platforms like Facebook, and even dating apps like Tinder.

The global recommendation system industry is poised for significant growth, with projections indicating a compound annual growth rate of 33 percent year-on-year until 2028. This would see the US market size surge from 2.29 billion USD in 2021 to an impressive 17.30 billion by 2028 [4], as illustrated in figure 1.

In light of this trend, it's crucial for businesses across all sectors, including financial institutions like Julius Baer to properly utilize recommendation systems to remain competitive. While such systems have traditionally been the domain of entertainment and e-commerce platforms, their potential in the banking and finance sector is vast. By adopting and integrating these systems, Julius Baer stands to offer financial products and services that are meticulously tailored to align with individual client preferences, needs, and financial behaviours.

Figure 1: Growth of Recommendation Systems in the US Market year-on-year

B. Problem Statement

The problem statement was initially proposed as creating a versatile recommendation system for use within Julius Baer. After interviews with various data scientists within the bank, the presence of existing recommendation systems was discovered.

An in-depth analysis of these recommendation systems revealed a few crucial flaws:

1. **Poor Personalization:** The systems lack personalization for the customers, being more inclined to recommend the most popular products rather than products that individual customers would likely enjoy. This can lead to generic recommendations that may not be relevant or appealing to the user, which could result in decreased engagement and lost sales opportunities for the bank.
2. **Lack of Adaptability and Monitoring:** Once the model was trained, it was used as-is, with no form of retraining or adaptation. Such a model becomes stagnant over time and, does not evolve with changing user behaviour or market trends. Without continuous updates or adaptability, the recommendations generated may become increasingly irrelevant or inaccurate, thus diminishing the system's effectiveness, and potentially leading to lost customer trust.

This is compounded by a lack of monitoring system through which the bank could track performance over time to spot any change in performance.

3. **Lack of User Interface:** There was an absence of a user-friendly interface for interacting with and interpreting the results of the recommendation systems. Without an intuitive and user-friendly interface, the bank may struggle to manage and optimize the system effectively, or understand how the recommendations are made.

Upon reviewing these findings with the lead data scientist under whom this project was overseen, the initial problem statement was redefined. The final problem statement was: Creating a versatile recommendation system with high levels of customer personalization & adaptability, a monitoring & automated retraining system, and an interactive user interface.

C. Solution Summarization

This paper details a comprehensive solution to enhance recommendation systems in Julius Baer. Central to the approach is the integration of the Light Gradient Boosting Machine Rank (LGBMRank) machine learning algorithm. This algorithm is designed to refine, rank and personalize recommendations, addressing a core issue of poor personalization observed in conventional systems.

To ensure the sustained accuracy and relevance of our recommendation model, a proactive monitoring and retraining strategy is implemented. This strategy involves continual assessment of the model's performance against a predefined threshold. When the model's performance falls below this threshold, an automatic retraining process is initiated using more recent data. This strategy ensures that the model remains aligned with evolving market conditions and customer preferences without the need for constant manual intervention.

Furthermore, to facilitate effective interaction with the recommendation system, a user-friendly dashboard using Streamlit was developed. This interactive platform allows stakeholders, including bank personnel, to explore customer data, observe the model's performance over time, and understand the rationale behind the product recommendations being made. This dashboard is designed to be intuitive and is aimed at bridging the gap between complex data analysis and actionable insights.

II. Literature Review

A. Machine Learning in Recommendation Systems

The potency of modern recommendation systems largely hinges on the capabilities of machine learning [1]. Machine learning, a subset of artificial intelligence, enables systems to automatically learn and improve from data without being explicitly programmed. In the context of recommendation systems, this translates to the ability to dynamically adapt to user preferences and behaviours, rather than relying on static, rule-based algorithms.

Machine learning empowers recommendation systems to:

1. Analyze and predict user preferences based on historical behaviour, and the behaviours of similar users, thereby suggesting items that users can more deeply resonate with.

2. Rank suggested items by relevance, ensuring that the highest quality recommendations are prioritized.

3. Quantify the probability of a user engaging with a particular recommendation, allowing for more targeted and effective suggestions.

Given the rapid advancements in the ML domain, it's anticipated that businesses will increasingly integrate sophisticated machine learning models into their recommendation systems. This evolution promises more personalized, accurate, and dynamic recommendations, further enhancing user experience and business outcomes.

B. Types of Recommendation Systems

Modern recommendation systems predominantly employ four primary techniques: User-User Collaborative Filtering, Item-Item Collaborative Filtering, Content-Based Filtering, and Hybrid Systems. Each of these techniques comes with its own advantages and disadvantages.

1) User-user collaborative filtering: User-user collaborative filtering operates on the principle of user similarity. It identifies users who share similar preferences with the target user, and recommends items that these similar users have shown interest in [5]. A graphical representation of this technique is shown in Figure 2. This approach can be encapsulated by the phrase, "Users like you also read A Dance With Dragons".

Advantages:

- Personalized recommendations based on user behaviour. These recommendations are tailored to each user, leading to a higher likelihood of positive interaction.

- No need for detailed item metadata. This reduces the requirement of extensive data collection about items.

Disadvantages:

- High computational complexity means scalability issues with growing user numbers [6]. As the user base grows, the system may become slow and expensive to run.

- New items and users have no interactions, and recommendations therefore can't be made. This is referred to as the "Cold Start" problem. It limits the system's ability to provide relevant suggestions for new users or items.

- Popularity bias means the system will be more likely to recommend popular items [7]. This can lead to a lack of diversity in recommendations, and a lack of personalization.

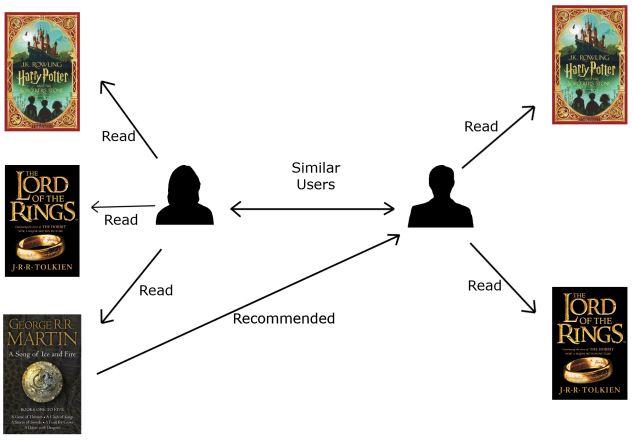

Figure 2: User-user collaborative filtering

2) Item-item collaborative filtering: Item-item collaborative filtering, in contrast, is a bit more elaborate, as it focuses on user-item interaction. It identifies items that the target user has engaged with, and recommends other items that are commonly liked by users who liked the initial item [5]. A graphical representation of this technique is shown in Figure 3. This approach can be encapsulated by the phrase, "Users who liked The Lord of the Rings also enjoyed Star Wars".

Advantages:

- Recommendations remain stable over time. This provides a consistent and reliable user experience.
- More scalable than user-user collaborative filtering [8]. Generally involves fewer computations as user and item count grows, making it more efficient.

Disadvantages:

- Requires detailed item profiles. This increases the complexity and costs of data collection and maintenance.
- Cold Start Problem for new items [9].

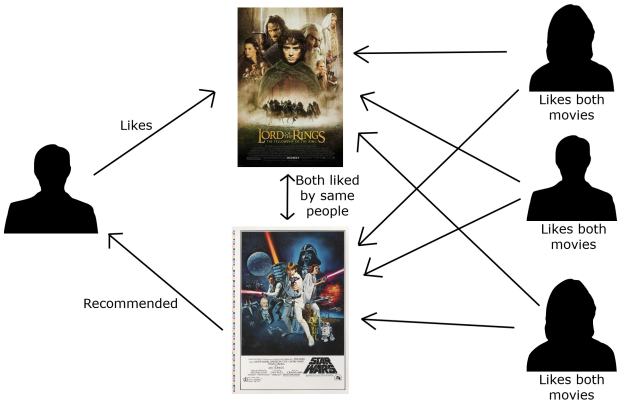

Figure 3: Item-item collaborative filtering

3) Content-based filtering: Content-based filtering delves into the attributes of items. Before making recommendations, this method requires a feature set for each item. User preferences are then deduced based on their interactions with these features [10]. A graphical representation of this technique is shown in Figure 4. This approach can be encapsulated by the phrase, "Since you watched action movies featuring Tom Cruise, we suggest Mission Impossible 7".

Advantages:

- Recommendations can be made for new users as long as initial preferences are provided. This helps to mitigate the Cold Start problem.
- Can recommend niche or less popular items. This introduces users to items they might not discover through other methods, referred to as "Serendipity".

Disadvantages:

- Requires extensive feature engineering [11]. Developing a meaningful set of item features can be labor-intensive and requires domain knowledge, increasing required resources to use it effectively.
- Risk of over-specialization for few things the recommendation system knows the user likes, referred to as a "Filter Bubble". This can limit the user's exposure to new and diverse content.

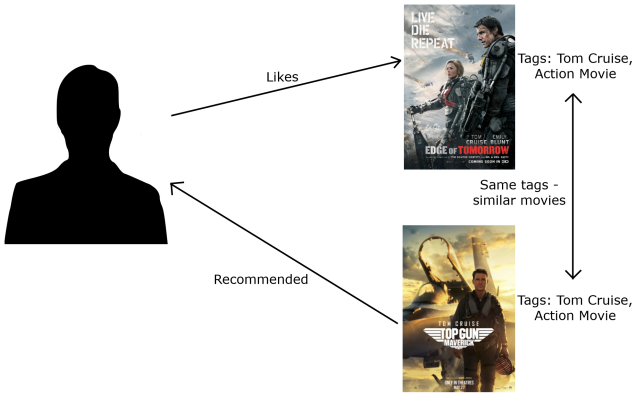

Figure 4: Content based filtering

4) Hybrid recommendation system: Hybrid recommendation systems combine the strengths of the aforementioned methods, offering a more holistic and adaptive approach. Given their versatility, they are the most popular recommendation system adopted in real-world applications [12].

Advantages:

- Combines strengths of multiple recommendation approaches. This allows for a more comprehensive and robust recommendation strategy.

- Often provides more accurate recommendations. By integrating various methods, it is possible to leverage their strengths for improved performance.

Disadvantages:

- Increased system complexity. Implementing and maintaining a hybrid system can be resource-intensive due to the integration of multiple approaches.
- Potentially higher computational costs. Combining multiple methods might lead to more intensive computation, which could affect system efficiency and cost.

III. Solution

A. Solution Structure

The development of the recommendation system for this project was systematically done in a stepwise manner, as illustrated in Figure 5. This section provides an overview of the solution structure, explaining the logic behind each stage of the system's construction:

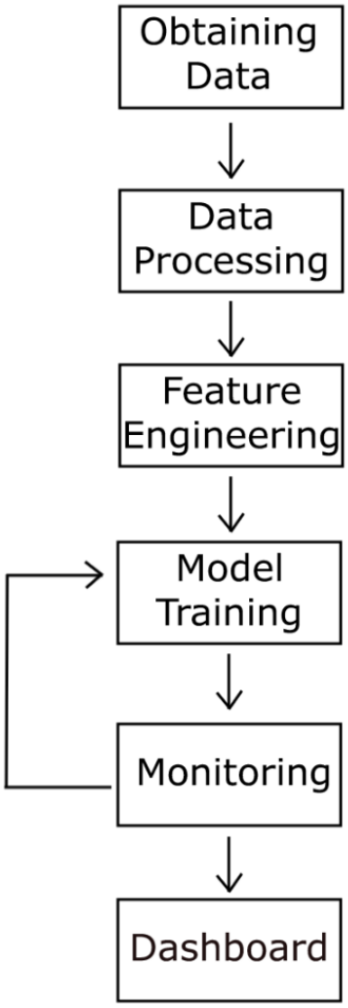

Figure 5: Process of Creating Recommendation System

1. **Obtaining Data:** The initial phase involved securing a relevant dataset. Due to privacy constraints, internal bank data was inaccessible. The search led to a Kaggle dataset from Santander, which closely mirrored Julius Baer's data. This dataset, therefore, formed the project's foundation.
2. **Data Processing:** The raw data underwent cleaning and preprocessing. Anomalies and inconsistencies, such as irregular missing value indicators, were identified and standardized to ensure data quality.
3. **Feature Engineering:** This stage involved creating new features from the existing data to increase the model's predictive power. Features were methodically designed to capture nuanced aspects of the temporal nature of the data, customer behaviour and product interactions.
4. **Model Training:** A user-user collaborative filtering approach employing cosine similarity was adopted. Following this, Light Gradient Boosting Machine Rank (LGBMRank) was integrated to refine and personalize the recommendations further. This approach directly addresses the problem of poor personalization, by using both collaborative filtering and machine learning techniques to generate personalized product recommendations based on individual customer profiles and behaviours.
5. **Monitoring:** A monitoring system was devised to ensure the model's performance over time. This system tracks the model's recall against a predefined threshold. If performance dips below this threshold, the model is retrained using more recent data, allowing it to remain attuned to evolving trends and customer preferences. This solves the problem of lack of adaptability and monitoring by introducing an automated, continuous check on the model's performance and initiating retraining when necessary, thus eliminating the need for manual model updates.
6. **Dashboard:** To facilitate data exploration and performance assessment, a visualization dashboard was created using Streamlit. This interface allows users to explore raw customer data, monitor the model's performance over time, and explore tailored product recommendations for individual customers. The creation of this dashboard directly addresses the lack of user interface, providing a convenient and intuitive platform for stakeholders to interact with and interpret the results of the recommendation system.

B. Data

1) Obtaining Data: The initial phase of the project was characterized by the challenge of securing a relevant dataset. As the bank was unable to provide data due to privacy concerns, a more local search was needed.

Within Julius Baer, the team this project was being collaborated on with was "Launchpad", the innovation lab. As a result, there were plenty of potential projects to work with, and create a recommendation system for. While the lab had many ongoing projects, the majority were either in their infancy, or lacked

the requisite data. An exception was the "Trend Radar" project, aimed at early trend identification to leverage emerging popularities. However, thorough exploratory data analysis revealed significant gaps in the data, rendering it unsuitable for building a recommendation system.

This led to the exploration of external datasets, specifically those that could align with the banking domain. The search culminated in the discovery of a dataset from a Kaggle competition sponsored by Santander, a Spanish bank [13]. The competition's objective was to design a recommendation system predicting a customer's next product purchase. After consultations with internal data scientists familiar with Julius Baer's data, it was clear that the Santander dataset bore significant resemblance to the bank's. This similarity implied that a recommendation system built on the Santander dataset could be adapted to Julius Baer's context with minimal modifications, and as such was selected as the dataset for the project.

2) Data Summary: The raw dataset tracked customer codes over a 17-month period, with monthly data points. A summary of the data is displayed in table 1. A key challenge was preserving the temporal nature of the data. Using a recommendation model that only considers a single month's data would strip away the valuable time-based patterns. To address this, the data was transformed into one-year windows. Each window encapsulated the most recent customer information, their product ownership, and other generated features that retained the temporal essence of the data.

Number of Data Points	13,647,309
Number of Features	48
Number of Unique Customers	956,645
Start Date	2015-01-28
End Date	2016-05-28

Table 1: Summary of the data

The dataset's features were broadly categorized into two segments: customer side, and product side. The former included attributes like age, gender, bank seniority, and income. The latter consisted of binary columns indicating whether a customer owned a particular product. These foundational columns were instrumental in generating new features that extracted additional information from the data.

The customer side data was one of the key elements in the personalization aspect of the recommendation system. In the existing Julius Baer recommendation systems, customer specific features are not retained or used. This was initially due to privacy concerns that have long since been dispelled. However, the systems have remained this way because the bank has determined that the effort required to overhaul them and implement additional customer level features is not worth it. Additionally, a potential security risk will always remain, and as such, the bank is reluctant in utilizing this data.

This project breaks free of these preconceived notions and limitations, and freely utilizes all available information in an attempt to demonstrate that leveraging customer level data is paramount in creating a more personalized recommendation system.

A significant portion of the dataset comprised categorical features. To ensure compatibility with a wide range of potential models, one-hot encoding was employed. This encoding method transforms categorical variables into multiple binary columns - a format that machine learning algorithms much better understand.

While one-hot encoding ensured model compatibility, it also expanded the dataset's dimensionality, making it more cumbersome and difficult to work with. This increased complexity became particularly evident when exploring user-user collaborative filtering techniques, which inherently have a computational complexity of $O(u^2 \cdot f)$, where u represents the number of users and f denotes the number of features used for similarity computation [14].

C. Data Analysis and Processing

Upon acquiring the dataset, the immediate priority was to conduct exploratory data analysis (EDA) to gauge its quality and suitability for the project. EDA is a crucial step in understanding the dataset's structure, identifying anomalies, and ensuring its readiness for subsequent modeling.

The EDA revealed a rich dataset, boasting millions of data points across a multitude of features. A silver lining was the minimal presence of missing values, a common challenge in many datasets. However, the data was not without its quirks.

A closer inspection during the cleaning phase unearthed inconsistencies in the representation of missing data. For example, while some entries used "NA" to denote a missing income value, others employed seemingly arbitrary numbers like "-999999". Such inconsistencies can skew analyses and lead to misleading model results.

To address this issue, a meticulous process of identifying and standardizing these irregularities was undertaken. A function was created that searched the dataset for these anomalous values, replacing them with the "np.nan" placeholder, a standard representation for missing data in Python [15]. Subsequent imputation strategies were then employed based on the nature of the data: categorical features were imputed with the mode (most frequent value), while numerical features were mean-imputed, factoring in the customer's province to retain some contextual relevance.

1) Observation and Behaviour Periods: In time-series machine learning projects, defining the observation and behaviour periods is a critical step.

The observation period is the time window where data is collected and used to create features for the machine learning model. This period is used to capture and summarize past behaviour or characteristics of entities based on which predictions are made. An illustration of this is available in figure 6:

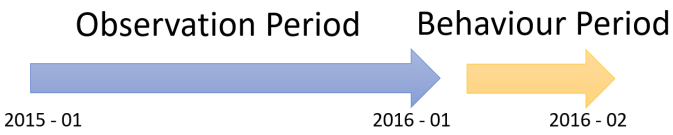

Figure 6: Observation period going from January 2015 until January 2016, and behaviour period lasting the following month

The behaviour period follows the observation period and is the time window during which the outcome, or behaviour, to be predicted is observed. This period is used to define the label or target variable for the machine learning model.

2) Applying Observation and Behaviour Periods to Data Splitting: Creating Train, Test, and Validation Sets: For this project, data splitting integrates the concepts of observation and behaviour periods into the division of data into training, validation, and testing sets. Ensuring a better evaluation of the model's performance under real-world conditions.

The observation and behaviour period split for the training and validation data is represented in figure 6, starting in January 2015 and ending in February 2016. The train and validation split was done by randomizing customer codes and selecting 80% of them as training data, and 20% as validation data. This split can be observed in table 2.

	Number of Customers
Training Data	17,592
Validation Data	4,416

Table 2: Number of Customers in Training and Validation Data

The test split, however, is structured to simulate a real-world scenario. To achieve this, the observation and behaviour periods are shifted by one month into the future, as illustrated in Figure 7.

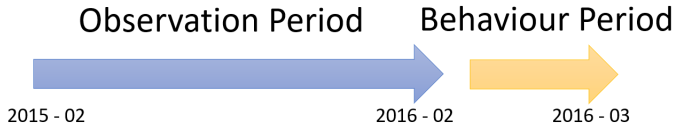

Figure 7: Observation period going from February 2015 until February 2016, and behaviour period lasting the following month

The testing period utilizes the most recent 11 months of the training and validation data's observation period, as well as the behaviour period, in order to evaluate performance. The new behaviour period is, likewise, shifted one month ahead, so as to provide unseen target data to test on.

This approach to data splitting, built around the concept of observation and behaviour periods, allows a robust evaluation of the model, providing insights into its potential performance once deployed in a live environment.

D. Feature Engineering

Feature engineering is a pivotal step in the data preparation process, often determining the quality of the subsequent model. It involves creating new features from the existing data to better capture underlying patterns, thereby enhancing the model's predictive power.

Before diving into feature creation, a comprehensive analysis of the existing features was undertaken. This involved categorizing them based on their nature (numerical or categorical) and their association (customer side or product side). Such categorization provided a structured view of the dataset, highlighting potential information gaps that the model might overlook.

Based on this analysis, several new features were conceptualized and generated to provide a better overall view of customer behaviour and product dynamics:

Customer Side Features

1. **Number of Products Owned:** This feature captures a customer's engagement level with the bank's offerings, potentially indicating their trust level or financial activity breadth.
2. **Product-to-Seniority Ratio:** By relating the number of products to a customer's seniority with the bank, this feature provides insights into their adoption rate and loyalty.
3. **Recent Product Acquisitions:** Tracking the number of new products a customer has acquired.
   (a) *Short Term (3 months):* Indicates recent changes in a customer's financial needs or growing relationship with the bank.
   (b) *Long Term (12 months):* Provides a broader view of a customer's product adoption pattern.
4. **Recent Product Drops:** Understanding the products a customer has relinquished.
   (a) *Short Term (3 months):* Signals recent dissatisfaction or changing financial circumstances.
   (b) *Long Term (12 months):* Gives insights into longer-term dissatisfaction or changes in financial situation.
5. **Product Selection Stability:** Evaluating how consistent a customer's product portfolio has been.
   (a) *Short Term (3 months):* Indicates recent financial stability and satisfaction with the bank's offerings.
   (b) *Long Term (12 months):* Provides a broader view of a customer's financial stability and satisfaction.

Product Side Features

1. **Average Customer Seniority per Product:** This feature can help in understanding the typical lifecycle stage at which customers opt for specific products.

2. **Product Popularity:** By counting the number of customers owning each product, the bank can gauge its popularity and potentially identify upselling or cross-selling opportunities.
3. **Average Customer Age per Product:** This provides insights into the age demographics attracted to specific products, enabling more tailored product design and recommendations.

Through these additional features, the dataset was enriched with nuanced insights, paving the way for a more informed and effective recommendation system. Such meticulous feature engineering ensures that the model is equipped with a comprehensive understanding of both customer behaviour and product dynamics.

E. Model Design

Before starting with creating the model, a primary recommendation technique was required. This was through process of elimination:

1. Item-item collaborative filtering does not make use of customer level data.
2. Content based filtering necessitates item level features to be generated. This requires in depth domain knowledge that the author does not posses.
3. Hybrid systems require at least two different recommendation systems before they can be implemented.

After this analysis, a user-user collaborative filtering approach, leveraging cosine similarity, was found to be the most fitting solution. The selection of cosine similarity as the primary metric for the collaborative filter was driven by its inherent strengths in certain areas. Cosine similarity excels in dealing with high-dimension datasets, a trait often seen in data used for recommendation systems [16].

Another benefit of cosine similarity is that it doesn't take zeroes into account when calculating similarity, further reducing computational complexity, especially when dealing with data that has been one-hot encoded. The metric's range, from -1 for complete dissimilarity to 1 for perfect alignment, also provides a straightforward, computer interpretable way to gauge similarity levels. The formula for cosine similarity is shown in equation 1 [17].

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||_2 \cdot ||\mathbf{B}||_2} \tag{1}$$

Where:

- $\mathbf{A} \cdot \mathbf{B}$ is the dot product of the vectors $\mathbf{A}$ and $\mathbf{B}$.
- $||\mathbf{A}||_2$ and $||\mathbf{B}||_2$ are the L2 norms (Euclidean norms) of the vectors $\mathbf{A}$ and $\mathbf{B}$, respectively.

Cosine similarity, however, is not without its limitations. The first of these is that it uses a linear approach, meaning it might miss out on more nuanced, non-linear user behaviour patterns [18]. Another disadvantage is that it doesn't consider relative values for each customer. For example, for a customer that only rates shows between 2 to 4 out of 5 stars, a 4 star rating is functionally equivalent to a 5 star rating for a customer that utilizes the full range of the rating system [19]. However, given that the data only consists of binary values representing product ownership, this disadvantage is a non-issue.

Before calculating cosine similarity between the primary customer and all other customers, the other customers must first pass a filter which checks that they own is at least one product that the primary customer does not own. This is done in order to improve computational time, as customers that don't have additional items won't provide any benefit for the recommendations, and the similarity computation will therefore be useless. All customers that make it through the filter are then ordered by similarity, and the top 1,000 most similar customers are selected. The model then returns a list of the most common products among the similar customers.

This is where the limitation of using a user-user collaborative filter becomes extremely apparent. As the data was millions of data points large, in order to have a reasonable run time, sampling was an inevitability. In order for the final model to run just one time within 30 minutes, a sample of 0.001 (0.1%) was required. This sampling amount was selected in an attempt to balance recommendation quality with running time.

For all intents and purposes, a pure collaborative filter would end at this point. However, unlike the real world, where the "true" selected products are unknown, the dataset being used contained a ground truth in the form of the subsequent month's product data. This allowed for the integration of a more sophisticated machine learning model to refine the recommendations further, and increase personalization.

This is where Light Gradient Boosting Machine Rank (LGBMRank) comes in. Tailored for ranking tasks, LGBMRank is a specialized variant of the Light Gradient Boosting Machine (LightGBM) [20]. It optimizes for ranking metrics, which aligns perfectly with the recommendation system's objective of maximizing personalization. One of the standout features of LGBMRank is that it is very good at handling sparse data, something especially useful when dealing with a one-hot encoded dataset.

Additionally, LGBMRank is designed to be efficient and scalable, making it practical for handling the large and complex datasets typical of the banking industry. It also accommodates the integration of categorical features, simplifying the preprocessing step significantly, thereby reducing the computational power required. While this is not a benefit that is taken full advantage of in this project, future implementations would greatly benefit from this. LGBMRank also ensures that the recommendations are not only relevant, but are also ranked in order of interaction probability, maximizing probable user engagement.

An illustration displaying the final model's design is presented below as figure 8.

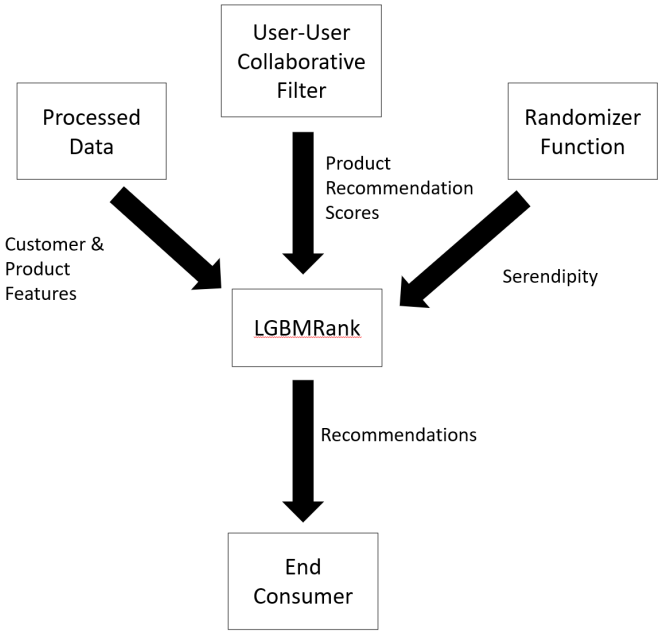

Figure 8: LGBMRank takes inputs from the processed data, the user-user collaborative filter, and a randomizer function

The LGBMRank takes various inputs, and produces recommendations for the end consumer as an output. The processed data provides the input for customer and product features. The user-user collaborative filter provides recommendation scores for each product, which is treated as an additional feature. Finally, a randomizer function provides some serendipity to the recommendations by substituting the recommendation the LGBMRank model is least confident in, with a random other product the consumer doesn't own, for 20% of the consumers.

F. Results

1) Performance Metric Selection: One of the last aspects of this project was to selecting a way to judge the quality of the recommendations, as well as creating a monitoring system. Evaluating the performance of a recommendation system is a nuanced task. Multiple metrics can be employed, from complex ones such as Mean Average Precision @ K, to simpler metrics like precision or recall. The formulas for these metrics are as displayed below. Recall is presented first, as equation 2.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2}$$

In this instance, recall measures the proportion of products that were actually purchased in a future month, which were recommended to the client.

As such, for this specific use case, it becomes:

$$\text{Recall} = \frac{RI}{RI + NR} \tag{3}$$

Where:

- RI is the number of correctly recommended items
- NR is the number of relevant items not recommended

Precision is a metric that quantifies the proportion of recommended items that are actually relevant to the user. In simple terms, it answers the question: "Of all the items the system recommended, how many were actually good recommendations?" The formula for precision is given as shown in equation 4.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{4}$$

For application in recommendation systems, this becomes:

$$\text{Precision} = \frac{\text{\# of Correctly Recommended Items}}{\text{Total \# of Recommended Items}} \tag{5}$$

Mean Average Performance @ K (MAP@K) is a measure that takes into account both the precision and the order of the recommendations. For each user, it calculates the average precision at various positions in the ranked list of recommendations (up to position K). Then, it averages this value across all users. In simpler terms, it answers the question: "How many of the top K recommendations are relevant, and how well are they ranked?" The formula for MAP@K is given as shown in Equation 6.

$$\text{MAP@K} = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{\min(m, K)} \sum_{k=1}^{K} \text{Precision@}k \cdot \text{rel}(k) \tag{6}$$

Where:

- Q represents the total number of users (or user queries) for whom the system is generating recommendations.
- m represents the number of items with which a user has interacted (e.g., purchased, clicked, liked) — these are considered as the 'relevant' items for that user.
- K represents the cutoff rank, which means the top K items recommended to a user are being considered. For example, if $K = 10$, the quality of the top 10 recommendations made to the user is being evaluated.
- Precision@k represents the precision calculated at each rank k, up to K.
- rel(k) is an indicator function that checks whether the item at rank k is relevant (i.e., whether a user has interacted with it). It is equal to 1 if the item is relevant, and 0 otherwise.

MAP@K provides a single number that tells us how good the recommendations are, both in terms of their relevance and their ranking. For a perfect recommendation system, MAP@K would be 1, indicating that all the top K recommended items are relevant and are perfectly ranked.

Given the nature of the dataset and recommendation system's design, recall was chosen as the evaluation metric. The recommendation system will always be recommending a fixed number of products. As this number is fixed and Julius Baer's intention is to capture as many of the true positives as possible, recall seemed the best alternative.

However, a challenge arose: most customers didn't acquire new products monthly. This meant that for a significant portion of customers, recall would always be zero, irrespective of recommendation quality. To address this, the recall calculation was refined to only consider customers who acquired new products in the following month. This adjustment ensured that the recall metric was not skewed by inactive customers. With this metric in mind, the next step was the creation of a monitoring system to use it.

2) Model Performance: Checking the results, a problem was identified with this project - that of overfitting. The LGBMRank model performs significantly better on training data than validation or test data. Typically, results range from a recall of 0.8-1.0 for training data, and between 0.42-0.8 for validation and testing data. Clearly, the model was overfitting. In an attempt to diagnose whether the problem stemmed from the model or the data, the model was changed to a Logistic Regression (LR) one. LR was chosen as the model to test with as interviews with internal data scientists confirmed that it is the ML model that is most frequently used in the existing recommendation systems.

Running the LR model, one thing became very apparent - the issue did not stem from the data. The results from the LR model were what was expected for - a train, test and validation recall of around 0.3-0.4. With this information, it became clear that the issue was with the LGBMRank model. Likely, the problem was to do with the hyperparameters, and tuning them would have solved it.

Despite the seeming improvement from the LR model, it had problems of its own. When checking the recommended products, it became evident that it simply always recommended the most popular products (savings account, guarantees, pensions, direct debit, and payroll). Considering this, tuning the hyperparameters of the LGBMRank model appeared to be the best choice for fixing the overfitting problem, as opposed to simply swapping out the model.

While hyperparameter tuning of the LGBMRank model was identified as the most promising solution to alleviate the overfitting problem observed in the results, the project faced significant constraints that made this solution difficult to implement. The use of user-user collaborative filtering in the recommendation system, as previously discussed, is computationally intensive, especially when performed on a laptop with limited processing capabilities.

This, coupled with tight time constraints, meant that there was insufficient opportunity to perform the rigorous hyperparameter tuning that the LGBMRank model likely required to optimize its performance. As such, the model kept was an LGBMRank model with default hyperparameters, as opposed to an LR model, due to the heavy popularity bias present in the latter.

3) Original Model Comparison: In order to gauge the improvement of the recommendation system created as compared with the original, new results were generated in an attempt to best replicate the original models used by the bank. The limitations are:

- All customer side information is removed. This limits personalization, and is much more similar to how the original bank data is structured.
- LR is used in place of LGBMRank. The use of LR models in the original bank's systems is something that was confirmed through interviews with data scientists at the headquarters in Zurich, and helps with the comparison
- The model retraining aspect has been limited through a flag in the original function. By disabling the model from retraining itself, it further emulates the original systems.

By implementing these limitations, the recommendation system has essentially been turned into one that the bank actively employs. Comparing performance between this variant of the system, and this project's one will allow for a numerical improvement to be calculated.

As discussed in the previous section, the system employing LGBMRank has an average training data of 0.8-1.0, with average validation and testing data of 0.42-0.6, occasionally achieving up to 0.8. Although it is clear that the model is overfitting, the performance has never dipped below a recall of 0.42 in tests or validation.

In comparison, the emulated original system does not overfit thanks to the use of LR, allowing its performance to be much more consistent. However, this consistent performance also generally performs worse than that of the project's system. An average test, train and validation recall of 0.35-0.4, as opposed to the aforementioned 0.42-0.6.

This difference represents anywhere between approximately a 5% to a 70% improvement over the bank's original model. With additional testing, this range could be narrowed down further. Despite the wide range of the performance increase, one thing is clear from this experiment - the original recommendation systems performance is not up to par with that of this project's.

In order to determine the potential value of the improved recommendation system, an estimate revenue calculation is performed.

- Assuming 1,000 customers pay attention to recommendations.
- Assuming 10% of these will purchase a recommendation that appeals to them.
- Assuming each additional service purchased nets the bank 5,000 USD a year.

- Assuming the true recall values are the average of the ranges of the original and improved model.

Original Model (Average Recall: 0.375)

$$\text{Original Successful Recommendations} = 1,000 \times 0.375 = 375$$

$$\text{Original Conversions} = 375 \times 0.10 = 37.5$$

$$\text{Original Revenue} = 37.5 \times \$5,000 = \$187,500$$

Improved Model (Average Recall: 0.51)

$$\text{Improved Successful Recommendations} = 1,000 \times 0.51 = 510$$

$$\text{Improved Conversions} = 510 \times 0.10 = 51$$

$$\text{Improved Revenue} = 51 \times \$5,000 = \$255,000$$

Revenue Difference

$$\text{Revenue Increase} = \$255,000 - \$187,500 = \$67,500$$

$$\text{Growth Percentage} = \left(\frac{\$67,500}{\$187,500}\right) \times 100\% = 36\%$$

This simplified estimation illustrates the potential revenue uplift with the enhanced recommendation system. With 36% increased revenue, this project's system is a considerable advancement over the original. Although this is a simplified example, it demonstrates the importance of a good recommendation system, and the potential benefits it can reap for Julius Baer.

G. Monitoring and Continuous Learning

The monitoring system is a rudimentary one that compares the test recall to a threshold set by the user. The system works by selecting a starting year and month, and training the model on the window from that date until one year in the future (for instance 2015-01 to 2016-01). It will then check the test recall, and compare it to the threshold selected. If the recall is surpasses or is equal to the threshold, it will shift the window one month in the future, and continue.

If the threshold is not reached, the model will retrain itself on the current window, in an attempt to improve recall. This is done taking into consideration that trends and customer preferences change over time, and a more recent model will likely better capture these shifts than an older one. If the threshold is still not reached after retraining, the model terminates with an error stating "Human Intervention Required". This is done in consideration of the fact that over time, the model may simply no longer be a good fit for the data, and must be changed.

This monitoring system not only helps keep tabs over how the model performs over time, but also allows for automated retraining, which helps with saving computational power, and increases uptime, as it is not retraining the model every for every new window. Tentatively, the threshold selected was a recall of 0.3, as this was determined to be a good midpoint between decent accuracy, while not being so high that the recall would fail to meet the threshold often.

Once the monitoring system reaches the final available date, it saves the results to a json file which is then used in the final portion of the project - a visualization dashboard.

H. Visualization Dashboard

In order to properly present the results, as well as present the data and performance in an easier to digest manner, a dashboard which presents some of the most important aspects of the project at once was also created. This dashboard was created using Streamlit, an application framework platform both user-friendly and versatile. The start page is displayed in figure 9.

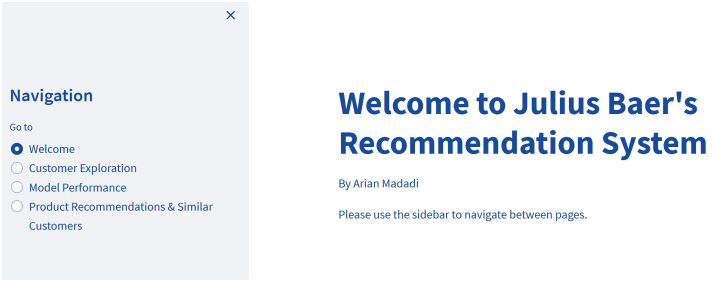

Figure 9: Dashboard welcome page

First and foremost is the raw data. In the "Customer Exploration" tab, users will be able to explore the raw customer data in order to more easily see what products each customer has over time. With the help of the drop down window shown in figure 10, users can select a customer code, and the page will display all the available information about that customer, across all dates. An example of this is shown in figure 11.

	grass_date	customer_code	employee_index	country_of_residence	sex	age	account_cr
0	2015-01-28	1,375,586	N	ES	H	35	2015-01-12
1,047,196	2015-02-28	1,375,586	N	ES	H	35	2015-01-12
1,681,276	2015-03-28	1,375,586	N	ES	H	35	2015-01-12
2,299,297	2015-04-28	1,375,586	N	ES	H	35	2015-01-12
2,714,121	2015-05-28	1,375,586	N	ES	H	35	2015-01-12
3,343,336	2015-06-28	1,375,586	N	ES	H	35	2015-01-12
4,581,564	2015-07-28	1,375,586	N	ES	H	35	2015-01-12
5,075,380	2015-08-28	1,375,586	N	ES	H	36	2015-01-12
6,258,167	2015-09-28	1,375,586	N	ES	H	36	2015-01-12
6,572,487	2015-10-28	1,375,586	N	ES	H	36	2015-01-12

Figure 11: Exploring data for customer code 1375586

This functionality not only streamlines the data verification processes, but also provides users with an intuitive interface to navigate and understand the dataset.

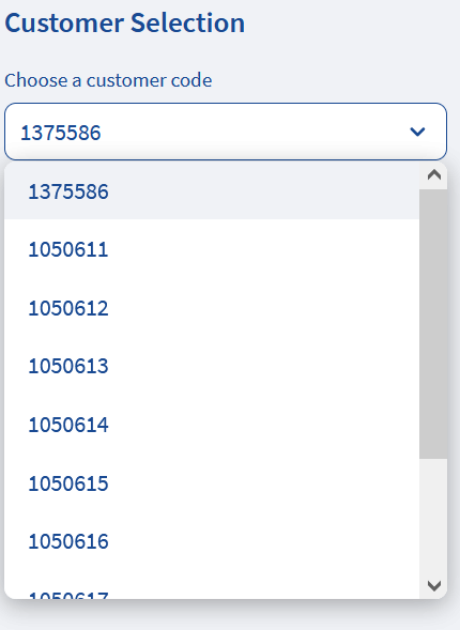

Figure 10: Customer Selection Drop-down

Another integral component of the dashboard is the 'Recall Over Time' graph, showcased in Figure 12. Selecting the "Model Performance" tab gives gives access to this section, where the visualization and data offer a temporal perspective on the model's performance. Any anomalies or significant shifts in performance can be quickly identified, providing insights into potential areas of improvement or external factors influencing the data, such as any temporal patterns the model is not capturing.

It's worth noting that unfortunately, the current dataset offers a limited scope, resulting in only three data points on this graph. However, as the model ingests real-world data, this visualization will grow, offering a richer understanding of performance trends.

The dashboard also offers tailored product recommendation insights for individual customers. Located in the "Product Recommendations & Similar Customers" tab, users can select a specific customer from a drop-down list which comprises the training group to promptly view their top product recommendations. As an illustrative example, the tailored recommendations for customer 271918 are showcased in Figure 13.

Enhancing the transparency of the recommendation system, the dashboard also provides a list of the top 10 customers most similar to the selected one. This feature not only underscores the logic behind product suggestions, but also empowers users to delve deeper. By taking these most similar customers, users can input them into the raw data exploration displayed in figure 11, allowing them to understand how and where the customers are considered similar.

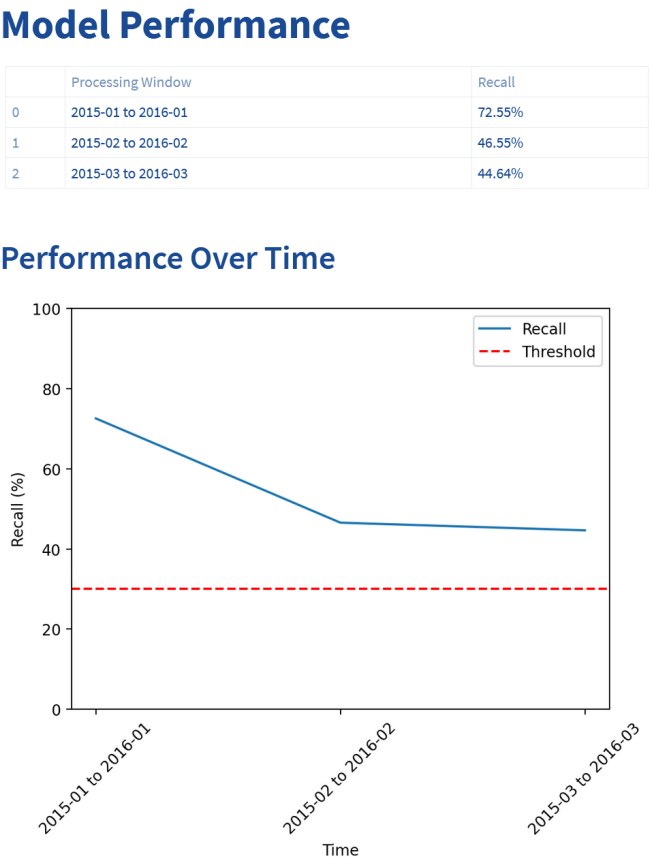

Figure 12: Model performance page

Figure 13: Tailored Recommendations and Similar Customers for customer 271918

While Streamlit offers an abundance of benefits, it does come with certain limitations. While it is one of the most straightforward and easy to implement application frameworks, it is not necessarily the most efficient. When working with large dataframes, it has a tendency to be quite slow. For instance, the customer code drop-down menu experiences slight delays, and not all customer codes are instantly accessible.

Additionally, generating the 'Recall Over Time' graph can be resource-intensive, especially when it tries generating the points on the spot. To remedy this, a system was where the model's results are first saved to a JSON file, which Streamlit then accesses to render the graph. These limitations come from the nature of Streamlit, and future optimizations, or simply a different application framework, would need to be employed in order to fully solve them.

IV. Discussion

The development of this recommendation system for use in Julius Baer's operational framework is not just a technological advancement for the bank; it's a new system that holds the potential to reshape the its customer engagement and increase revenue streams.

Firstly, at the heart of any financial institution is its client base. By providing more tailored recommendations, Julius Baer is enhancing the customer experience. A more personalized banking experience means customers feel understood and valued. This not only fosters loyalty, but also increases the likelihood of customers adopting more of the bank's products and services. Every recommendation that translates into a product purchase or service adoption, directly contributes to the bank's revenue.

The tangible financial benefit of this improved recommendation system is evident from a basic projection. The simplified estimation suggests a potential revenue uplift of nearly 36%. This not only signifies the system's potential profitability but also underscores the importance of recommendation systems in Julius Baer's growth strategy. Additionally, this rough revenue calculation only represents the tangible benefits from increased recommendation quality, and does not take into account other benefits of this project's recommendation system, such as the visualization dashboard.

The automated monitoring and retraining system embedded within the recommendation pipeline is there to ensure the quality of the model outputs not only in the present, but also in the future. By ensuring that the model remains relevant and accurate, the recommendation system is future-proofed, allowing Julius Baer to use it for longer periods of time before requiring inspection. This approach not only ensures consistent performance, but also reduces the long-term costs associated with manual monitoring and frequent model overhauls and retraining. This cost-saving, when combined with the potential revenue from increased model adaptability further increases the bank's profitability.

Additionally, the dashboard, tailored exclusively for internal bank members, serves an important role in the recommendation system's ecosystem. Its interactive nature offers a comprehensive view, allowing bank employees to delve deep into the underlying logic behind each recommendation, as well as viewing its performance over time. This allows the employees to be more confident in the recommendations that each customer is being given. It also serves as a tool for internal teams to refine strategies. By understanding customer behaviours and preferences, marketing and product teams can design more effective campaigns and product offerings, further driving revenue.

Finally, the project was made with the idea of versatility in mind. While increasing personalization and implementing a feedback system were paramount, another key aspect of the project was creating a system that could easily be iterated upon and changed in order to fit many different recommendation system applications within the bank. As such, the project is made with many separate files, classes, and functions, so that it is clear what each component does, and what needs to be changed can easily be identified and swapped out. Whenever a recommendation system will be needed for a future project, the foundation will exist, minimizing the amount of work required to create exactly the kind of system needed.

V. Further Work and Conclusion

1) Further Work: While the current system offers significant advantages over the existing ones being adopted by the bank, there are several avenues for improvement and expansion:

1. **Enhanced Feature Engineering:** The current model's performance, though promising, has room for improvement. Modern recommendation systems often employ extensive feature engineering to extract every piece of information from the available data. By diving deeper into the data and generating more features, the system's predictive accuracy can be enhanced. Exploring advanced techniques such as embeddings or temporal patterns could unlock previously untapped potential in the data.

2. **Diversifying Recommendation Approaches:** The exclusive reliance on user-user collaborative filtering, while effective, has its limitations. Incorporating item-item collaborative filtering or content-based filtering can lead to a more comprehensive hybrid recommendation system. This would not only help address the cold start problem, but it would also potentially enhance the quality of recommendations. The system could also implement matrix factorization or deep learning models to add more robustness, diversifying the sources of recommendations and potentially achieving better accuracy.

3. **Hyperparameter Tuning:** The overfitting problem observed with the LGBMRank model suggests the need for hyperparameter tuning. Systematic hyperparameter optimization, such as grid search or Bayesian optimization, can help to improve the model's performance. Given adequate computational resources and time, tuning hyperparameters could be the key to resolving the overfitting issue, and significantly improving the model's effectiveness.

4. **Continual Learning and Feedback Integration:** Modern recommendation systems tend to have continual learning, adapting to new data and feedback. By integrating a way for customers to provide feedback on recommendations, the system could refine its algorithms and improve recommendations. This feedback loop, both implicit (clicks, views, purchases) and explicit (ratings, reviews), could serve as a valuable source of data for model refinement.
5. **Enhanced Performance Evaluation Mechanisms:** While the current monitoring system is effective, a more granular monitoring system with additional performance evaluation metrics can provide insights into specific areas where the model might be underperforming. Additionally, introducing anomaly detection systems can highlight sudden drops in performance, and allow for proactive troubleshooting before significant issues arise. This would allow for more timely interventions and adjustments, better ensuring the model remains at peak performance at all times.
6. **Balancing User Experience and Profitability:** Currently, the recommendation system gives recommendations purely to maximize user experience, with little regard for anything else. Considering that different services and products the bank offers provide different returns, it is imperative to carefully consider these differences when generating recommendations, in order to strike an optimal balance between user satisfaction and financial performance.
7. **Integration with Other Systems:** The recommendation system can be further integrated with other bank systems, such as customer relationship management (CRM) tools or marketing platforms. This would not only allow for a more holistic view of the customer, but may even lead to more tailored recommendations.
8. **Additional Interpretability Features in Dashboard:** The current dashboard effectively visualizes model performance and product recommendations. However, the only interpretability it offers is a list of the most similar customers to the selected one. Adding additional features that offer deeper insights into the reasoning behind each recommendation could substantially enhance user trust and engagement. Incorporating tools like Local Interpretable Model-Agnostic Explanations (LIME) or SHapley Additive exPlanations (SHAP) into the dashboard would provide granular insights into the model's decision-making process, offering a clear breakdown of the driving factors behind each recommendation.

Incorporating these changes will not only likely improve the system's performance, but also ensure that Julius Baer remains at the forefront of personalized banking experiences. The ultimate goal is to create a system that not only meets, but anticipates the needs and preferences of the bank's clientele, driving both customer satisfaction, interaction, and revenue.

2) Conclusions: In the end, this project's contributions to Julius Baer were threefold:

1. A versatile recommendation system with increased personalization.
2. A monitoring system that automates model retraining and ensures performance quality remains consistent.
3. A user-friendly dashboard that provides intuitive displays from the model performance and recommendations.

The first major contribution was the development of a versatile recommendation system that significantly enhances personalization. By integrating the LGBMRank algorithm, the system generates more personalized and appropriate recommendations. This level of personalization enables Julius Baer to offer better financial advice, aligning the bank's services more closely with each customer's unique needs and preferences, and potentially deepening customer engagement and trust. This increase in personalization can be quantified as a potential 36% increase in revenue derived from successful recommendations. Additionally, the recommendation system is written in a style such that it is easily modified for more niche and specified recommendation tasks.

The second key contribution is the implementation of a robust monitoring system that automates the process of model retraining. This continuous monitoring ensures that the model's performance quality remains at a consistently high level. This automation significantly reduces the need for constant manual oversight and ensures that Julius Baer can respond quickly and effectively to changing customer behaviours and market conditions.

The third and final contribution is the creation of a user-friendly dashboard that offers intuitive and transparent displays of both model performance and product recommendations. This tool empowers bank personnel to interact directly with the system, enabling them to gain insights into customer preferences and the effectiveness of the recommendation engine. With this dashboard, Julius Baer not only enhances its internal decision-making processes, but also establishes a foundation for greater transparency and collaboration between the bank and its clients.

These contributions collectively position Julius Baer at the forefront of innovation in private banking. They equip the bank with the tools and agility necessary to navigate the ever-evolving financial landscape, where responsiveness to customer preferences and market dynamics is paramount.

REFERENCES

[1] L. R. B. S. Francesco Ricci, *Introduction to Recommender Systems Handbook*. 2010. [Online]. Available: https://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf.
[2] J. E. Solsman. "Youtube's ai is the puppet master over most of what you watch." (2018), [Online]. Available: https://www.cnet.com/tech/services-and-software/youtube-ces-2018-neal-mohan/.
[3] Netflix. "Decoding the defenders: Netflix unveils the gateway shows that lead to a heroic binge." (2017), [Online]. Available: https://about.netflix.com/en/news/decoding-the-defenders-netflix-unveils-the-gateway-shows-that-lead-to-a-heroic-binge/.

[4] G. V. Research, "Recommendation engine market size, share trends analysis report by type (collaborative filtering, hybrid recommendation), by deployment, by application, by organization, by end-use, by region, and segment forecasts, 2021 - 2028," 2020. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/recommendation-engine-market-report/.

[5] M. D. Deepjyoti Roy, "A systematic review and research perspective on recommender systems," 2022. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00592-5.

[6] P. C. Pradeep Kumar Singh Shreyashee Sinha, "An improved item-based collaborative filtering using a modified bhattacharyya coefficient and user–user similarity as weight," 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10115-021-01651-8.

[7] D. S.-M. et al., "Social influence-based similarity measures for user-user collaborative filtering applied to music recommendation," 2019. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-99608-0_30.

[8] C. I. et al., "Pointer-based item-to-item collaborative filtering recommendation system using a machine learning model," 2021. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0219622021500619.

[9] F. R. Mahamudul Hasan, "An item–item collaborative filtering recommender system using trust and genre to address the cold-start problem," 2019. [Online]. Available: https://www.mdpi.com/2504-2289/3/3/39.

[10] M. A. A. Yassine Afoudi Mohamed Lazaar, "Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network," 2021. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1569190X21000836.

[11] S. H. N. et al., "A brief analysis of collaborative and content based filtering algorithms used in recommender systems," 2021. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/981/2/022008/meta.

[12] M. H. et al., "A recommender system for complex real-world applications with nonlinear dependencies and knowledge graph context," 2019. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-21348-0_12.

[13] B. Santander. "Santander product recommendation." (2016), [Online]. Available: https://www.kaggle.com/competitions/santander-product-recommendation.

[14] C. Maklin. "Memory based collaborative filtering — user based." (2022), [Online]. Available: https://medium.com/@corymaklin/memory-based-collaborative-filtering-user-based-42b2679c6fb5.

[15] H. C. et al. "Numpy." (2023), [Online]. Available: https://numpy.org/.

[16] J. A. K. Michael D. Ekstrand John T. Riedl, "Collaborative filtering recommender systems," 2010. [Online]. Available: https://experts.umn.edu/en/publications/collaborative-filtering-recommender-systems.

[17] F. Karabiber. "Cosine similarity." (2010), [Online]. Available: https://www.learndatasci.com/glossary/cosine-similarity/.

[18] T. P. Tan Thongtan, "Sentiment classification using document embeddings trained with cosine similarity," 2019. [Online]. Available: https://aclanthology.org/P19-2057/.

[19] R. H. S. et al., "Movie recommendation system using cosine similarity and knn," 2020. [Online]. Available: https://aclanthology.org/P19-2057/.

[20] G. K. et al., "Lightgbm: A highly efficient gradient boosting decision tree," 2017. [Online]. Available: https://papers.nips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.