Optimizing Product Recommendations for a Try-Before-You-Buy Fashion E-Commerce Site

Paul Ruh

Dissertation written under the supervision of professor Joren Gijsbrechts

Dissertation submitted in partial fulfilment of requirements for the MSc in Business Analytics, at the Universidade Católica Portuguesa, 02.01.23.

ABSTRACT

Title: Optimizing Product Recommendations for a Try-Before-You-Buy Fashion e-commerce Site
Author: Paul Ruh

The fashion e-commerce market has experienced a significant growth and more and more customers tend to buy products online, rather than in physical stores. However, after a customer buys a product online, only a fraction of the garments stay in their wardrobe as many items are being returned to the vendor. Due to the absence of physical examination and misleading product descriptions customers struggle to find the right product suitable to their personal preferences. Especially the category of women's lingerie suffers to a great extend from high return rates. Different sources report that between 70 up to 100% of women wear wrong sized bras. Personalized recommendations through so called recommendation systems play an essential role in e-commerce. This thesis aims to optimize the current product recommendations of a Belgium start-up called CurveCatch that sells women's lingerie articles online and relies on a try-before-you-buy concept. To predict which products a customer is likely to buy two different personalized deep learning approaches were introduced. Data sparsity was addressed by labeling each unique product per customer and minority classes were synthetically oversampled. The findings demonstrated that recommendation systems are not only relevant for companies operating on a large scale. Rather, they also can be a valuable source of accurate recommendations for start-ups with sparse data. However, results also underlined well-known limitations of recommendation systems. Both models struggled especially when identifying products a customer is likely to buy, while it was rather easy to identify products a customer is not likely to buy.

Keywords: Machine learning, Recommendation systems, Deep learning, Data sparsity, Fashion, E-Commerce, Product Recommendations

ABSTRACT

Título: Optimização das recomendações de produtos para um site de comércio electrónico Try-Before-You-Buy Fashion
Autor: Paul Ruh

O mercado de e-commerce de moda experimentou um crescimento significativo e cada vez mais os clientes tendem a comprar produtos online, em vez de em lojas físicas. No entanto, muitos itens são devolvidos ao vendedor após a compra online, pois os clientes têm dificuldade em encontrar o produto certo adequado às suas preferências pessoais devido à falta de exame físico e às descrições de produtos enganosas. A categoria de lingerie feminina sofre muito com as altas taxas de devolução. Diferentes fontes relatam que entre 70% e 100% das mulheres usam sutiãs do tamanho errado. As recomendações personalizadas através dos chamados sistemas de recomendação desempenham um papel essencial no e-commerce. Esta tese visa otimizar as atuais recomendações de produtos de uma start-up belga chamada CurveCatch que vende artigos de lingerie feminina online e depende de um conceito de experimente antes de comprar. Para prever quais produtos um cliente é mais propenso a comprar, foram introduzidos dois diferentes abordagens de aprendizado profundo personalizadas. A escassez de dados foi abordada rotulando cada produto único por cliente e as classes minoritárias foram sobreamostradas sinteticamente. Os resultados demonstraram que os sistemas de recomendação também podem ser uma fonte valiosa de recomendação de produtos para start-ups com dados escassos. No entanto, os resultados também sublinharam as bem conhecidas limitações dos sistemas de recomendação. Ambos os modelos lutaram especialmente ao identificar os produtos que um cliente é mais propenso a comprar, enquanto era relativamente fácil identificar os produtos que um cliente não é propenso a comprar.

Palavras-chave: Aprendizagem de máquinas, Sistemas de recomendação, Aprendizagem profunda, Espaçamento de dados, Moda, E-Commerce, Recomendações de produtos

02 INTRODUCTION

2.1 Introduction to CurveCatch

This thesis aims to supplement and improve the current operational processes of the Belgium lingerie retailer CurveCatch by developing decision support models that identify products customers are likely to buy. The firm relies on a so called **try-before-you-buy (TBYB)** concept, which allows customers to try on various lingerie products at home before they pay. As shown in the graphic below, the company gathers information about individual user features by a small questionnaire that all customers are obliged to fill out before items can be sent to their home. After a customer filled out the survey a style expert recommends several products, which will be presented to the client in a so called **"virtual box"**. In a virtual box the customers select desired items online to be passed to their home in a **"Box"** without being charged for the products. After trying on the items at home, the customers decide which products they wish to keep and which products they want to return. If customers would like to order another box, they can decide if they want to fill out the survey again or if the virtual box should be based on the previous answers of the questionnaire.

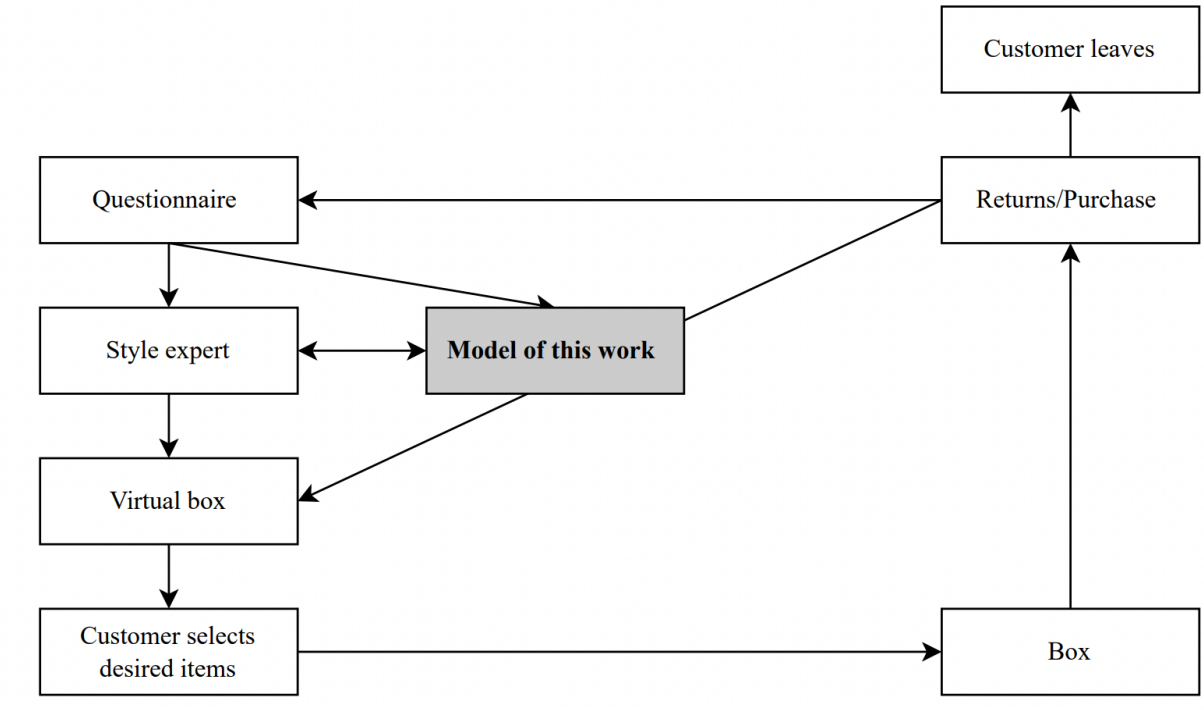

Figure 1: Business approach of CurveCatch

Currently, recommendations that ultimately end up in a virtual box are made by style experts, which is costly and time-consuming. This work focuses on partially replacing **the human-in-the-loop** component of the current process by developing a top-line recommendation system to produce a ranked list of items per customers. As shown in the graphic above, this work is positioned right next to the style expert and aims to supplement the current process of making accurate product recommendations. It is important to note that at this stage of the start-up it is not the goal to fully replace the style expert but to build a valid starting point to select items suitable for the virtual box.

Producing the ranked list is the top line of the recommendation system of CurveCatch. Other decision support models may determine which products should be part of the virtual box by building up on the probabilities from the ranked list of products and many other constraints like profit margins or inventory levels. Ultimately, this work can be seen as the top part of the recommender system of CurveCatch with significant relevance as many other models will build up on the results. Recommender systems have played a major role in e-commerce for a long time. This research will introduce several machine learning models to tackle well-known limitations of recommender systems. As mentioned before, the main objective of this research is to identify products customers might like by producing a ranked list of products per customer, while the final selection of which products to position in the virtual box remains out of scope.

2.2 Research problem

Techniques to derive ranked list of products are widely known. There exists a large variety of models and techniques backed-up by profound literature of well-known research institutes and established companies like Amazon or Netfilx (Hunt, 2015; G. Linden B. S., 2003). For this work it is crucial to find the right approach and architecture specific to the available data and individual challenges of CurveCatch. While companies like Amazon or Google have extremely large amounts of data available, start-ups like CurveCatch can only utilize significantly smaller amounts which aggravates the challenge a lot. On the one hand this thesis aims to choose a valid machine learning architecture which suits the individual problem and approach of CurveCatch and on the other hand this work tries to find a way to successfully derive insights out of sparse data. Hence, 2 research questions were identified:

RQ 1: *How can machine learning be leveraged to find out which products a customer is most likely to buy?*

Two personalized deep learning models will be developed which involves training a model on labeled data to predict whether a customer will purchase a particular product based on features such as the customer's demographics and the purchase history. Relevant user features from the questionnaire and several product details serve as input features for the neural networks while the label indicates whether an article was purchased or not. The model classifies all products in the inventory of CurveCatch and outputs a probability of purchase for each of the items.

RQ 2: *How can the data sparsity be addressed to still get results with high confidence?*

Data sparsity will be addresses by labeling each unique product per customer to extend the historical orders data frame. An oversampling technique will be introduced to create synthetic samples of the positive classes. Finally, the model architecture and hyperparameters need to be adapted to perform well on sparse data.

03 RELATED WORK

3.1 Recommender systems in E-commerce

The global E-Commerce market has been constantly growing, especially during the pandemic online sales have increased (Anam Bhatti, 2020). Along with this increase, the shopping behavior of users has changed, as more and more people tend to buy fashion products online. However, after a customer buys a product online, only a fraction of the garments stay in their wardrobe as many items are being returned to the vendor which in turn reduces profitability and increases the carbon footprint (Nestler, 2021). Customer returns have risen sharply in the last years with growth yearly growth rate of 50% for certain categories of garments (Choi, 2016). The reasons for returns vary from “Just tried it on for fun” to misleading product descriptions. Yet, the most frequently occurring reason for returns in the fashion industry is related to the size of the article (Hannu Saarijärvi, 2017; Ratcliff, 2014). The purchase decision of clients involves individual factors like the taste of the individual and product related factors like the color, the design, or the fabric of product. Additionally, the physical examination of the product is absent when shopping for products online. Thus, the aforementioned factors lead to major uncertainties in the online fashion industry and ultimately return rates grow (Nestler, 2021). Users are frequently confronted with an overwhelming selection of goods and their corresponding descriptions due to the web's tremendous rise in information. Thus, personalized recommendations enabled through so called “recommender systems” (RS) have been playing and essential role to boost sales and to facilitate the decision-making process of the client (Shuai Zhang, 2019).

Especially the category of women’s lingerie suffers to a great extend from high return rates. Different sources report that between 70 up to 100% of women wear wrong sized bras, as the traditional method of bra sizes often overestimates band size and underestimates cup size (A.R. Greenbaum, 2003; K. Wood, 2008; Steele, 2010; J. White, 2012). Especially for the TBYB-approach of CurveCatch it is key to make solid product recommendations based on the gathered data from previous orders and from the questionnaire. Currently most recommendations are made knowledge-based by fashion experts. Once the start-up will scale-up, manually recommending products will get more costly and time-inefficient. Like many other companies CurveCatch can significantly boost efficiency and sales by developing recommender systems to efficiently support

users in the buying process by designing information agents that point to products likely to be purchased by a client (Burke, 2007).

Typically, recommender systems are based on user preferences, item features and past interactions between a user and an item. Recent literature describes a recommender system as a "search ranking system, where the input query is a set of user and contextual information, and the output is a ranked list of items" (Heng-Tze Cheng, 2016). RS aim to predict whether a particular user is likely to buy a product or not, which means deriving the probability of purchase between a specific user-item pair. (F.O. Isinkaye, 2015). In most cases the ranked list of products is backed-up by other decision support models which also consider constraints like profit margins of products and inventory levels (Long-Sheng Chen, 2008).

To model preferences of users, the first step is to obtain sufficient representative data or feedback of the respective user (Gawesh Jawaheer, 2010). Here, a distinction between explicit and implicit feedback is made. **Explicit feedback** requires active user participation to gather the desired data. Often a customer is asked to rate a product after the purchase. Many companies like Netflix or Amazon ask their customers to rate bought products on a scale from one to five to better understand preferences of customers and to ultimately improve their product recommendations (Gawesh Jawaheer P. W., 2014). **Implicit feedback** is automatically tracked and monitored by a system. This includes tracking which items a user bought and on which items the customer clicked (Lei Chen, 2021). When comparing explicit with implicit feedback, explicit feedback is thought to be a more reliable foundation for product recommendations (F.O. Isinkaye, 2015). In the case of CurveCatch the amount of explicit information about the user they can rely on is large, as all customers are obliged to answer a questionnaire that examines both style and size preferences. On the other hand, the accessible implicit feedback about the order history is sparse for the young start-up. In comparison with larger companies, it is important to note that the implicit data available to CurveCatch is significantly smaller. Fundamentally, this represents a disadvantage which can be compensated by the deep knowledge about the user. A detailed description how the explicit feedback can be leveraged to balance the sparse implicit feedback can be found in section 4.2.

Additionally, to the differentiation between the type of collected data many ways to produce a ranked list of items for a specific customer are available. The two most common techniques for recommender systems are:

1) Content-based: Mainly focuses on the attributes of the product (Pankaj Agarwal, 2018). Mostly suitable for products like books with an extensive product description where naïve machine learning methods like Bayes classification are common. Content-based filtering methods examine the description of items the user has rated and the description of products that potentially could be recommended (Justin Basilico, 2004). The problem is most often handled as a "user-specific classification problem" (Burke, 2007) and tries to predict whether a user is likely to be interested in an unseen product. Training instances consist of products the user bought, rated, or found interesting. The filtering system selects products based on the relationship between unseen items and items the user has reacted positively to, which can also be called the accumulated information about user based on explicit feedback. In a simplified example this means: When a user buys a shirt of the brand Nike, the content-based filtering system can recommend other shirts of the same brand since many product features are similar.

2) Collaborative-based: Utilizes relationships of past ratings, e.g., purchases or likes, by identifying peer users and forming "a weighted vote over these neighbors" to predict if a customer is likely to buy an unseen product (Justin Basilico, 2004). Ultimately collaborative filtering methods try to predict a rating a user would give to an article he or she never interacted with based on the ratings of similar users. Practically this means that the model aims to fill gaps of a sparse matrix where each user is represented as rows and each item is represented as columns where purchases or ratings are the values of the matrix. Companies like Amazon build their recommendation systems based on "item-to-item collaborative filtering" methods. They recommend products that seem similar to the products the client actually purchased and that have been frequently bought together. Here a common challenge to overcome is the "cold-start-problem". In a nutshell: What to do with new users that did few ratings? (G. Linden, 2003). Stitch Fix, a personal styling service that uses recommendation systems to provide personalized clothing recommendations to its users in a TBYB approach is one of the main competitors of CurveCatch. They also leverage collaborative filtering mechanisms to rank and order all items in their inventory. In some way the problem is simplify able as: Those who have liked what customer C liked, also

liked product P. So, product P would be a suitable recommendation for customer C (Stitchfix, 2022).

Start-ups, like CurveCatch, often have a small and rapidly changing customer base due to limited resources, rapid growth or changing market situations, making it difficult for both collaborative-based filtering and content-based filtering to accurately recommend products or services. Additionally, both types of recommendation systems rely on a large amount of data to make accurate recommendations, and start-ups may not have sufficient data available to effectively use these types of systems (Demiriz, 2004). Therefore, CurveCatch may be better served by using other types of recommendation systems, such as hybrid systems that combine different methods or personalized systems that consider individual preferences and characteristics.

In many cases both collaborative and content-based filtering have merged to a hybrid architecture to achieve better results (Pankaj Agarwal, 2018; Justin Basilico, 2004; Prem Melville, 2002). (Burke, 2007) has identified 7 different Hybridization techniques, like weighting different model, switching between models depending on confidence scores or mixing the predictions. Recently, deep learning has gained more and more popularity in the domain of recommender systems due to its ability to solve more complex problems while providing higher precision and recall. With the rise of deep learning the model architecture of modern RS has changed drastically to capture non-linear relationships between items and users based on more complex representations of data. In recent years, a variety of different advanced deep learning models have emerged including both supervised & unsupervised machine learning models, attention-based models, and several models within computer vision (Shuai Zhang, 2019). However, this work solely focuses on supervised machine learning models as the label to predict is clearly defined and known. All in all, the two main factors that contribute to the widespread use of deep learning recommenders today are nonlinear transformation by activations such as relu or sigmoid and advanced representation learning through embeddings or one-hot encoding.

The success of this work is closely tied to selecting the right machine learning architecture from the multitude of available possibilities to build a top-line recommendation system. It is key to derive insights from the extensive user features CurveCatch has accumulated through the survey.

As described above, conventional filtering systems are solely based on past ratings or purchases which seems not suitable to the general conditions the Belgium start-up provides. Rather, it is necessary to develop a personalized architecture that utilizes user features gathered through the questionnaire.

3.2 Common limitations of recommender systems

Recommender systems have become an integral part of many online platforms, from e-commerce websites to streaming services. While they have the potential to improve the user experience and increase engagement, they are not without limitations.

One of the most widespread problems is called the **cold-start problem**, which occurs when the system is unable to make recommendations for a new user or a new item that has not been rated by other users. The problem is related to insufficient information about user and item features due to data sparsity. Often, when a new product is introduced to the inventory or when a new customer is acquired RS suffer setbacks in the quality of the recommendation (Blerina Lika, 2014). This problem happens when implicit data on purchase behavior of customers is sparse or missing. For example, a collaborative filtering mechanism would never recommend a product that not yet has been rated or purchased because there is not enough data on users' preference available (Andrew I. Schein, 2002). Another major limitation is the so-called **popularity bias**, which refers to the tendency of the system to recommend the most popular items or items with the highest ratings, regardless of the individual preferences of the user. Popularity bias often emerges when the distribution of the data has a long tail, which is also the case for CurveCatch. This means that few items reflect most sales and therefore popular items are dominating the recommendations. There are several ways to tackle popularity bias, among them adjusting the distribution the prior training or performing a post-hoc re-ranking (Yang Zhang, 2021).

To address these limitations, researchers have proposed various solutions. For example, some have suggested incorporating additional data sources, such as social media data or demographics, to provide a more comprehensive view of the user and their preferences (O'Donovan J., 2014). Others have proposed using hybrid systems that combine the strengths of different types of algorithms (Y. Koren, 2009). Another tool to improve recommender systems are heuristics. These simple, efficient rules or mental shortcuts can help users make decisions quickly and accurately. For example, the

"rule of thumb" heuristic suggests that a recommender system should prioritize items that are popular or well-rated by other users. Additionally, it is possible to narrow down the sheer volume of available options by providing a quick and easy way to make informed decisions (Deuk Hee Park, 2012).

04 MODEL CHOICE & JUSTIFICATION

This section aims to construct the conceptual models that ultimately will be applied to the data and if possible to mathematically translate the concept. The goal of this work is to generate a ranking of all products per user with a corresponding probability of purchase. The data from the survey can be perceived as user features while the data on past purchases offer product features as well as implicit data about the history of purchases. The main model of this work is a deep learning recommender that predicts the label $\boldsymbol{y_{ui}} = \{\ \mathbf{1}\ (\boldsymbol{purchase/box}); \mathbf{0}\ (\boldsymbol{no\ purchase})$ in a binary classification problem. To extend the scope of the model and to zoom into the results the label has been enlarged to $\boldsymbol{y_{ui}} = \{\ \mathbf{2}\ (\boldsymbol{box\ and\ purchase}); \mathbf{1}\ (\boldsymbol{box\ b.no\ purchase}); \mathbf{0}\ (\boldsymbol{not\ chosen})$ in a second model, where "box b. no purchase" indicates a recommendation of the style expert which has not been purchased by the customer, and "box and purchase" indicates that indicates a recommendation of the style expert which has been purchased by the customer. The table below provides a detailed description of all labels:

Label	Description
1	The style expert didn't choose the product for a recommendation. Therefore, the user-product pair had no interaction yet.
2	The style expert recommended the product, and the customer chose the article in a virtual box to be passed to his or her home. However, he or she did not decide to buy the item
3	The style expert recommended an article, the customer chose it in a virtual box to be sent home. Additionally, the customer decided to buy the item.

Figure 2: Label description

Class imbalance was tackled with an oversampling technique called SMOTE while both product and user features were extracted using logistic regression. All categorical features were one-hot encoded while continuous features have been normalized to scale from 0 to 1. Finally, 330 input features were fed into 4 dense layers with relu as the activation function with shapes of 330, 160, 80 and 3. In the first model the final layer consists of a Sigmoid function with the shape of 1, which outputs the final probability of purchase. In the second model, the final layer consists of a Softmax activation with a shape of 3 to output three different probabilities, each of them specifying the probability of a item belonging to a class.

As already highlighted in section 3 of this work modern recommender systems often consist of more complex deep learning architectures as they simply outperform more simple techniques. The data of CurveCatch is sparse and just relying on traditional collaborative filtering methods would not result in good model performance. Rather, it is more important to exploit the extensive knowledge about the user CurveCatch has through the questionnaire every new customer is obliged to fill out. Considering the sparse implicit data and the high number of relevant user features a deep learning model seems a promising choice for the current data and will gain even more relevance when the start-up will scale up and more purchases are available as implicit feedback. Although a simple model like a logistic regression or a random forest might perform better now, it is crucial to develop an approach that will help the start-up in the long term. The following section provides an in-depth description of the model, where every step is explained separately. First, the overall classification problem is explained by formulating the models, and second, class imbalance, features selection, and feature extraction are explained.

4.1 Model formulation

Essentially, the classification problem of CurveCatch can be assessed in two different ways. Let *U* symbolize the number of unique users and *I* the number of unique items. Ultimately, the goal is to predict the label y_{ui}, which indicates if a specific user bought or didn`t buy an item after the style expert recommended some articles to the customer. However, when considering the classification only as a binary classification problem where $y_{ui} = \{ 1\ (purchase/box); 0\ (no\ purchase)$, the model would miss out on important information as it is not distinguishing between a box and a purchase. Instead, the classification problem can be considered as a multi-label-classification problem with 3 labels in total. In other words, instead of mimicking what experts are currently doing the model leverages all implicit information available in the historical orders data. To extend the scope of this work both the binary classification and the multi-label classification will be analyzed and compared.

Multi-label classification	***Binary classification***
$\boldsymbol{y_{ui} = 2}\ (box\ and\ purchase)$	$\boldsymbol{y_{ui} = 1}\ (purchased/box)$
$\boldsymbol{y_{ui} = 1}\ (box\ but\ no\ purchase)$	$\boldsymbol{y_{ui} = 0}\ (not\ purchased)$
$\boldsymbol{y_{ui} = 0}\ (not\ chosen)$	

Furthermore, for profound research, it is crucial to define what to expect from a well-performing model and which metrics characterize good performance. Accuracy might be a first indication of how well the model performs, but especially when working with sparse data other metrics like precision and recall are more informative. The precision-recall trade-off is a fundamental concept in the field of machine learning, particularly in the context of classification tasks (Buckland, 1994). It refers to the trade-off between the precision and recall of a classifier, where precision is the proportion of positive predictions that are actually correct, and recall is the proportion of actual positive instances that are correctly predicted. In other words, precision measures the accuracy of positive predictions, while recall measures the completeness of the predictions Particularly in the Case of CurveCatch, it is of high relevance to discuss the trade-off between precision and recall before deploying the model. Before passing items to the customer's home the client chooses items in a virtual box according to personal preference. Even if none of the items were selected by the customer only small to little costs arise for the firm as shipping costs are avoided. In this case, having more false positive predictions would be more tolerable than missing out on true positives, which means that recall seems to be more relevant than precision while precision still can´t be ignored.

This can be further underlined by the visual representation of the expected value framework below. Let M_p denote the profit margin and S_{cp} the shipping costs of a specific product P. The graphic below shows that significant costs for CurveCatch only occur after the item is shipped to the customer.

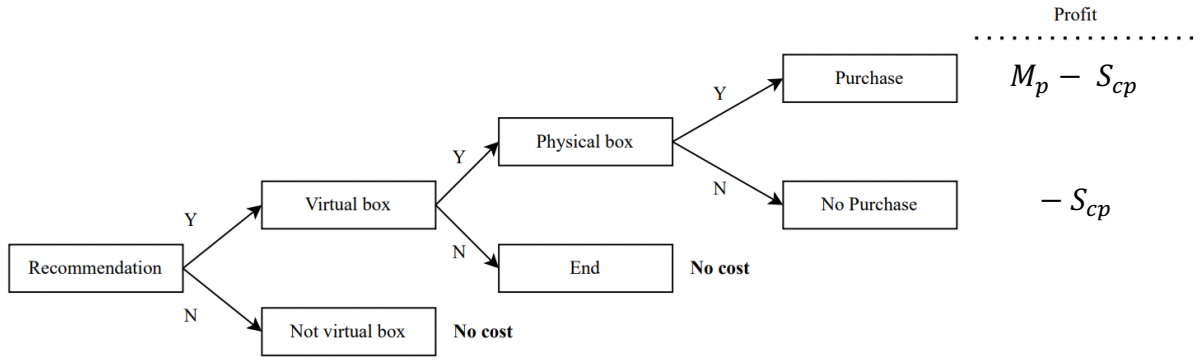

Figure 3: Expected value framework

Finally, the expected value $E(X)$ for the event of recommending a product to the customer could be formulated as follows where $P(B)$ denotes the probability that a customer is going to buy a specific product conditional:

$$E(R = Y) = P(B \mid R = Y, S = Y, B = Y) * \left(M_p - S_{cp}\right) - P(B|R = Y, S = Y, B = N) * (S_{cp})$$

The expected value framework underlines that costs only arise after customers chose articles virtual by themselves in a "virtual box". In other words, before shipping costs might arise a style expert has to recommend a product to the client in a virtual box $(R = Y)$ and the customer has to select the article from the virtual box $(S = Y)$. Only after both conditions happen to be true, shipping costs arise for CurveCatch. This highlights again that Type 1 errors seem to be more tolerable than Type 2 errors, which is crucial when defining a classification threshold. Ultimately, it is of high relevance to correctly identify all samples of the positive classes. Otherwise, the model simply misses out on possible sales which would contribute to the recommendations of CurveCatch in a negatively. This underlines that recall seems more important than precision and that the classification threshold should be set lower than a default value of 0.5.

4.2 Feature selection & engineering

When predicting which fashion articles customers are likely to buy, style and size are important features to consider (Hollander, 2018; P. Kaur, 2020). Style refers to the overall design and aesthetic of the article, such as its color, pattern, and silhouette (Fry, 2017). Size refers to the dimensions of the article and how it will fit on the body (K. Chung, 2021). As mentioned before, every unique product will be labeled per customer with the labels "box and purchase", "box but no purchase" or "not chosen". There are ways to approach this problem: i) Label all unique products per customer and disregard size. Consequently, every product will only occur once per customer ii) Label all unique product for each size that is available. For instance, product X would be labeled five times for the sizes A, B, C, D, E. For this work, option one has been chosen. This means the model purely analyzes how well the style suits a specific customer. If the product only occurs once the model can't consider all possible sizes, otherwise every product would occur multiple times for each available size. This means that the model is heavily focused on finding the right style for a specific customer. Nevertheless, the actual size is not the only size feature the data of CurveCatch offers. The questionnaire asks customers about their desired circumference, cup size and band size

which all have been included to the model as size features. Additionally, customers are asked to indicate their breast position and the firmness of their breasts to find the most suitable bra. This can be especially important when finding the right product category, as for instance women with space in between their breasts might prefer a bralette over a plunge.

Additionally, it is important to make a clear distinction between user & product features. Product features refer to characteristics or attributes like price, brand, category, and color. User features, however, are mostly composed of what customers state in survey. For example, the above-mentioned breast position, the favorite brand and the age of a user. Out of the whole data set ten user features and seven product features have been selected as input features for the model.

Selecting those features that are most useful for predicting the label is a key challenge when setting up supervised machine learning models. Too many features can confuse the model and too few features may miss out on important information in the data. Manually running a logistic regression to analyze the relationship between features and target variable seems logical, however, also seems inefficient and time-consuming due to the high amount of input features. Instead, it makes more sense to utilize an algorithm called Recursive Feature Elimination (RFE). RFE starts with all variables in the train data set and removes features that might lead to inaccurate results. It is possible to choose several machine learning models that compute a "measure of variable importance", in this case, a logistic regression was chosen. This means according to the coefficient and the significance level an importance score is calculated to eliminate the least important features. Afterward, the model will be refitted until a desired number of features is achieved (Max Kuhn, 2018). There are two important hyperparameters to select: i) the desired number of features and ii) how many features should be removed in each iteration. As the features were already pre-selected there is no need to remove any of them. After trial and error, the model achieved the best results without any further feature elimination. Both tables below show the final features that have been defined as input features for the deep learning models. Finally, all categorical features have been one-hot encoded, and all continuous features have been normalized which results in 333 input features.

User features

Feature name	Type	Description
Circumference	Categorical	European circumference of the bra the customer wears most often
Brand	Categorical	Favorite brand of the customer
Cup size	Categorical	European cup size of the bra the customer wears most often
Filling	Categorical	How much padding/filling does the customer like to wear
Breast position	Categorical	Description of firmness of breasts
Sure right size	Categorical	How sure is the customer that he/she knows the size of the bra you wear most often?
Underwire	Categorical	Perforation in terms of underwire/ breace
Firmness	Categorical	Description of position of breasts
Willingness to pay	Continuous	How much is the customer willing to pay in €?
Age	Continuous	Age in years of the customer

Figure 4: User feature description

Product features

Feature name	Type	Description
Vendor	Categorical	The vendor of the product
Product type	Categorical	Product type (Bra, swim wear)
Color	Categorical	Color of the product
Cup size	Categorical	European cup size of the product
Band size	Categorical	European band size of the product
Model type	Categorical	Model of the bra (e.g. full cup, plunge, slip…)
Price	Continuous	Price in € of the product

Figure 5: Product feature description

When comparing user and product features with each other it becomes apparent that many user features match product features. For example, the price of the product matches the willingness to pay of a customer and the favorite color of the client matches the actual color of the product. In a

way, the model learns to map user features to the product features to find suitable products for a recommendation. For instance, if a customer is willing to pay a maximum amount of 50€ for an article the model could simply exclude all products above that price.

4.3 Tackling class imbalance

The training data ends up in a **severe class imbalance** due to the low number of purchases and recommendations. With 99% of all class labels being $y_{ui} = 0$ the model will struggle to find user & product preferences that characterize a purchase. This means, without any further data manipulation the model would learn quickly how to correctly predict the negative class (true negatives) but will encounter difficulties when predicting the positive class, which would result in very few true positives, low precision, and low recall. The problem of class imbalance is widely known, and many papers suggest different techniques to oversample the minority class or to undersample the majority class. Often hybrid techniques of over- and undersampling yield promising results (Khoshgoftaar, 2019). However, when a product is labeled as $y_{ui} = 0$ the style expert chose to not recommend the product to the customer justified by expertise. Thus, undersampling the majority class would eliminate relevant information about the negative class in the data. Additionally, after testing different techniques to tackle class imbalance oversampling has proven to yield significantly better results than undersampling or hybrid techniques. To synthetically generate more samples of the minority classes a technique called SMOTE (Synthetic Minority Over-sampling Technique) has been chosen. SMOTE selects examples in the feature space that are near to one another, draws a line connecting the examples, and then creates new samples at positions along the line (Nitesh V. Chawla, 2002).

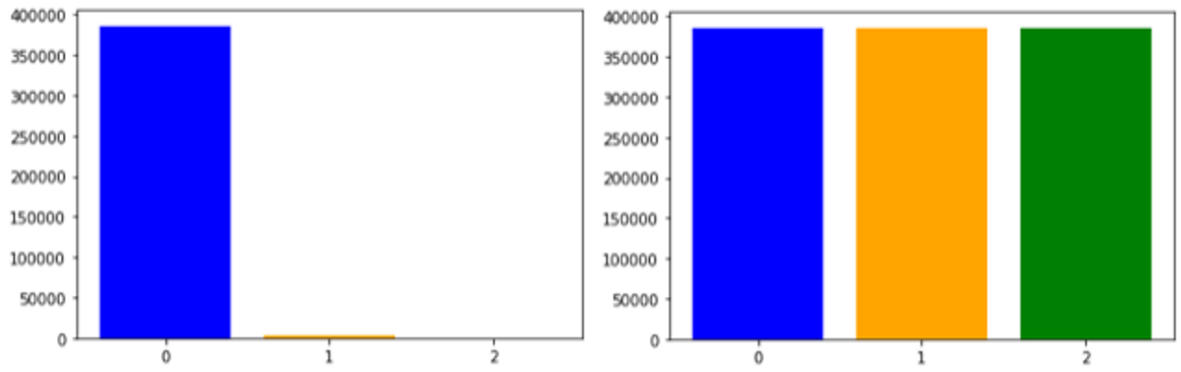

Figure 6: Oversampling of multi-label model

The training data of the multi-label classification problem ends up having 386,327 samples labeled as 0 (“not chosen”), 275 samples labeled as 1 (“box”) and 464 samples labeled as 2 (“purchase”). Both minority classes were oversampled to the same number as the majority class which resulted in all labels having 386,327 samples. The bar charts below show the balance of the training data both before and after oversampling.

The same procedure has been followed for the binary classification problem where the label 0 has been oversampled from 3,259 samples to 386,327 samples.

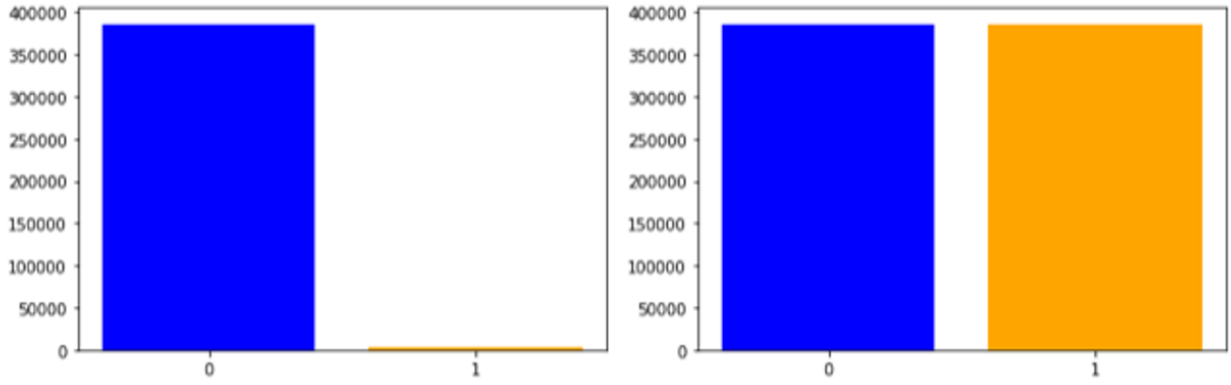

Figure 7: Oversampling of binary classification model

04 RESULTS

In this section first the binary classification will be evaluated separately in Section 4.1. To extend the scope of the model and to zoom into the results the multi-label classification model will be evaluated and compared to the binary classification model in Section 4.2.

4.1 Binary classification problem

The training, validation, and testing set has been split randomly, where training instances consist of 80% of all instances and validation and training consist of 10% of all samples respectively. As the model has to handle sparse data the number of dense layers is reasonable. All 333 input features have been reduced to 160 in a first dense layer and consequently reduced to 80 in a second dense layer. All dense layers relied on relu as the activation function, the loss was calculated with binary cross entropy and adam was selected as the optimizer which offers the benefit of not manually selecting the learning rate of the model. After 10 epochs the gap between validation and training accuracy was minimized which indicates that 10 epochs are a good choice as a hyperparameter. The validation accuracy of the model varied between 83% and 87%, with a small variance. The validation loss of the model varied between 43% and 34%, with a slightly higher variance. In terms of class-specific performance, the model had a precision of 100% and a recall of 84% for class 0, indicating strong accuracy in identifying instances of this class. However, the model had a precision of only 3% and a recall of 52% for class 1, indicating poor accuracy in identifying products that have been purchased or already recommended by a style expert of CurveCatch. The following section provides an in-depth analysis and interpretation of all results. On average, the predicted probability of purchase for label 1 was approximately 0.44, compared to a probability of 0.19 for label 0. This already indicates that the model found some characteristics to distinguish between both classes and that samples of label 1 on average end up higher in the ranking then samples of label 0.

4.1.1 Accuracy and loss

The left plot below shows the accuracy of the binary classification model after ten epochs while the right plot shows the loss. It is noticeable that the validation accuracy starts high and doesn’t change much in later epochs. Additionally, the gap between validation and training accuracy seems relatively small which indicates that overfitting is not present and that the model should generalize relatively well on unseen data. With over 42% the validation loss also starts relatively high in the

first epoch but decreases to 36% in the last epoch. For both the validation accuracy and the validation loss variance is visible.

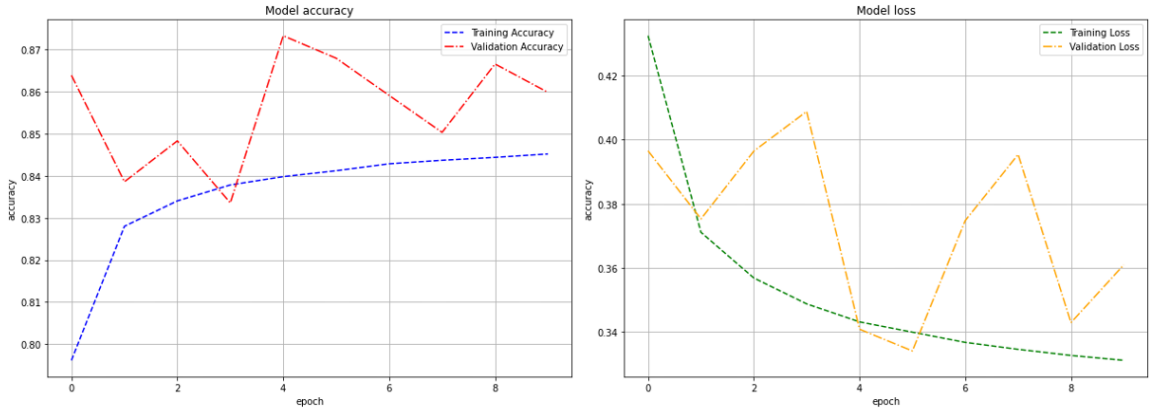

Figure 8: Accuracy and loss of binary classification model

Variance in validation accuracy refers to the degree to which the validation accuracy of a machine learning model fluctuates or changes over time. High variance in validation accuracy can indicate that the model is unstable or sensitive to small changes in the training data, which can be a problem because it can make it difficult to accurately assess the model's performance. It can be difficult to define a specific threshold for what is considered high variance in validation accuracy, as it can depend on a variety of factors, such as the complexity of the model, the nature of the task being performed, and the quality of the training and validation datasets. In general, however, high variance in validation accuracy is typically considered to be any fluctuation or change in validation accuracy that is significantly larger than what would be expected given the inherent noise and variability in the data. One way to quantify variance in validation accuracy is to calculate the standard deviation of the validation accuracy over multiple training epochs. If the standard deviation is relatively large compared to the mean validation accuracy, it can be an indicator of high variance (Alpaydin, 2014). The standard deviation for the validation accuracy (σ_{va}) equals 1.15 on average and the standard deviation for the validation loss (σ_{vl}) equals approximately 3. Both σ_{vl} and σ_{va} seems acceptable compared to the mean validation loss and accuracy. It is worth noting that there is no hard and fast rule for what constitutes high variance in validation accuracy and loss, and the acceptable level of variance can vary depending on the specific context and goals of the machine learning task. In general, however, it is generally desirable to have low variance in validation accuracy in order to build stable and reliable machine learning models, which in this

context is given. This states that the model is not too complex for the given task and overfitting should not be present.

4.1.2 Precision, Recall, Confusion matrix and AUC-ROC

Accuracy is a useful metric for evaluating the performance of a neural network, but it is not the only metric that should be considered, because accuracy alone does not provide a complete picture of how well the neural network is performing. **AUC-ROC (Area Under the Receiver Operating Characteristic curve)** is a useful metric for evaluating the performance of a deep learning model when the target variable is binary (Fawcett, 2006). The ROC curve plots the true positive rate (also known as recall) against the false positive rate at different classification thresholds. AUC-ROC provides a single metric that summarizes the performance of the model across all classification thresholds, with a higher value indicating better performance. One reason to consider AUC-ROC when evaluating a deep learning model is that it is insensitive to class imbalance because it is based on the ranking of predicted probabilities, rather than the actual class labels (Flach, 2012; J.A. Hanley, 1982). This is especially important to CurveCatch, as the data ends up in a severe class imbalance before oversampling.

Furthermore, **a confusion matrix** is another useful tool for evaluating the performance of a deep learning model. It shows the number of true positive, true negative, false positive, and false negative predictions made by the model, and can help you understand where the model is making mistakes. By understanding the types of errors the model is making, areas where the model may lack become identifiable (Powers, 2011; Stehman, 1997).

Figure 9 below shows the confusion matrix for the binary classification approach. When analyzing the **confusion matrix**, the high number of true negatives stands out at first glance. This means that the model is effective at correctly pinpointing items that customers are not likely to buy. Nevertheless, it is also noticeable that the model identified a significant number of samples in the test set as products a customer is not likely to buy when they actually were labeled as 1, which implies the product was either purchased by the customer or recommended by the style expert but not chosen to be purchased by the client. On the other hand, the model identified an almost equal number of true positives compared to true negatives (~400). This means that approximately half of all potential items for a box or purchase have not been classified correctly. This ultimately means

opportunities to generate revenue for CurveCatch will be overlooked. When considering the **ROC curve, the model** compared to a baseline value of 0.5, which represents a random classifier, performs significantly better. However, performance decreases with a true positive rate (TPR) and false positive rate (FPR) greater than 0.8. There are a few potential reasons why the performance of a deep learning model might decrease with high TPR and FPR. One possibility is that the model is overfitting to the training data, and is therefore not generalizing well to new, unseen data. This can cause the model to make overly confident predictions that may not be accurate. Another possibility is that the model is not adequately considering the context or characteristics of the input data. For example, if the model is making predictions based on a limited set of features, it may not be able to accurately capture the complexity of the data. In the case of CurveCatch the data set only included 3.259 instances of the positive class which have been oversampled to ~386k samples. This might indicate that the data set doesn't consist of enough samples labeled as 1 to effectively represent features that characterize a purchase or an item worth putting into a box.

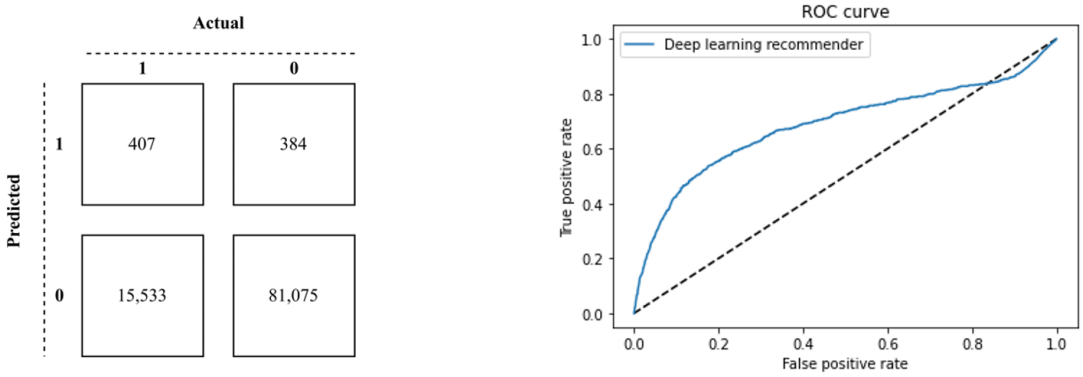

Figure 9: Class-specific metrics - Binary model

By looking at the values in the confusion matrix, it is possible to calculate the precision, recall, and F-score for each class, as well as for the overall performance of the model. Precision and recall are two evaluation metrics that are commonly used in the fields of information retrieval and machine learning. Precision measures the proportion of items that are correctly identified as positive by a classification model among all items that are identified as positive. On the other hand, recall measures the proportion of positive items that are correctly identified by the model among all positive items in the dataset. The F-score, also known as the F1 score, is a metric that combines precision and recall into a single measurement. It is calculated as the harmonic mean of precision

and recall (Christopher D. Manning, 2008; Mark D. Smucker, 2007). The trade-off between precision and recall has already been discussed in section 4.1. To recap, recall has been identified as a more relevant metric to evaluate performance while precision shouldn't be fully ignored. The table below shows all three metrics along with the support samples from the test set.

	Precision	**Recall**	**F1-Score**	**Support**
0	1.00	0.84	0.91	96608
1	0.03	0.52	0.05	789

Figure 10: Preccision & Recall binary classification

Based on the results above, it appears that the model performs well at identifying class 0, with a precision of 100% and a recall of 84%. However, the model has low performance when it comes to detecting class 1, with a precision of only 3% and a recall of 52%, which underlines the findings derived from the confusion matrix. The high precision for class 0 indicates that the model is very accurate at identifying instances of class 0 (not purchased), and that a large proportion of the instances it classifies as class 0 are actually class 0. However, the model is missing a significant number of instances of class 0 (~16%). On the other hand, the low precision for class 1 implies that the model is not very accurate at identifying class 1, and that a small proportion of the instances it labels as class 1 are actually class 1. Taking into account that false positive predictions don't necessarily lead to additional costs for CurveCatch a low precision seems acceptable, yet still indicates room for improvement. A recall of 52% indicates that approximately half of all products that might lead to a purchase have not been correctly classified. This underlines the conclusion drawn from the confusion matrix that a lot of potential revenue was not identified correctly. Overall, these results suggest that the model may need further improvement, particularly in its ability to accurately identify class 1. One possible approach to improving the model's performance could be to collect more training data for class 1, or to fine-tune the model's hyperparameters. Improvements and discussions will be discussed in depth in section 5.4.

4.1.3 Optimizing the classification threshold

A useful tool to evaluate the performance of binary classifiers and to optimize the classification threshold are precision-recall curves. The precision-recall curve plots precision on the y-axis and

recall on the x-axis for different thresholds of the classifier's predicted probability (Christopher D. Manning, 2009).

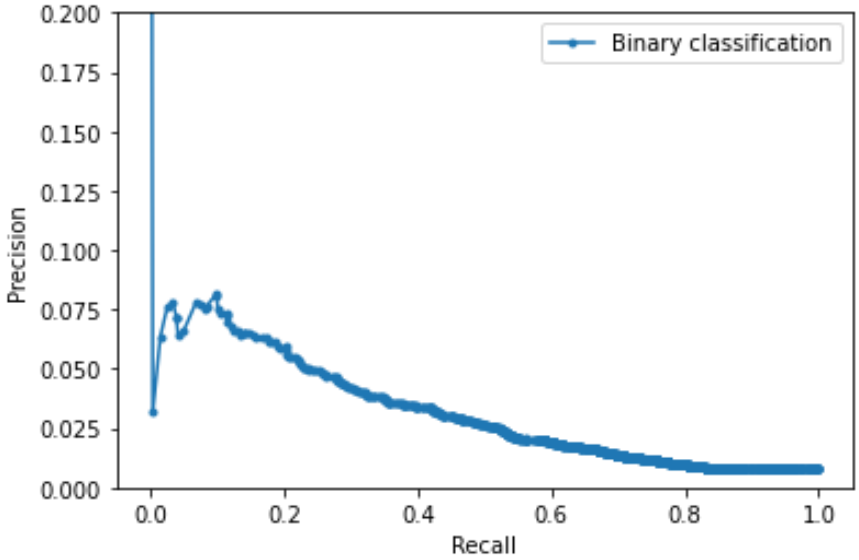

Figure 11: Precision-recall curve

Figure 11 above shows the precision-recall curve for the binary classification model. As mentioned in the previous section, setting the cutoff to 0.5 the model yields a precision of 3% and a recall of 52%. The plot indicates that small trad-off in precision can result in a high increase in recall. For instance, if the classification threshold will be adjusted so that the model returns a precision of 1% instead of 3%, the recall would increase to approximately 70%. However, only analyzing metrics for the positive class doesn't draw the full picture of the trade-off. By decreasing the classification threshold results for label 0 suffer as well. The table below shows precision, recall and f1-score for different classification thresholds.

	Label 1			Label 2		
Cutoff	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
0.5	0.03	0.52	0.05	1	0.84	0.91
0.4	0.02	0.54	0.04	1	0.80	0.89
0.3	0.02	0.59	0.04	1	0.76	0.86
0.2	0.02	0.66	0.03	1	0.68	0.8
0.1	0.01	0.72	0.02	1	0.58	0.7

Figure 12: Threshold optimization

With a cutoff of 0.1 the recall of label 1 can be increased to 72%, however, the recall of label decreases from 0.84 to 0.58. Considering that the mean predicted probability of class 1 was 0.44 compared to 0.19 for class 0, the optimal classification threshold should lie somewhere in the middle. With a threshold of 0.25, the recall of label 1 increases to 61% while the recall of class 0 decreases to 73%, which represents a justifiable trade-off and a significant improvement of the model. In general, CurveCatch can find a higher number of true positives by decreasing the classification threshold. After analyzing the precision-recall curve and the table above it is suggested to set the classification threshold to 0.25 which would yield the results shown in the table below:

	Precision	**Recall**	**F1-Score**	**Support**
0	1.00	0.73 (-11%)	0.84 (-7%)	96608
1	0.02	0.61 (+9%)	0.04 (-1%)	789

Figure 13: Precision & Recall of binary classification with new threshold

4.2 Extension to a multi-label classification problem and comparison

To recap, the multi-label classification model predicts the labels = {0: no purchase; 1: box but no purchase; 2; box and purchase} and outputs three columns of probabilities, each of them being the probability of a user-product pair belonging to one of the three classes. The model architecture is similar to the binary classification approach; however, the last layer consists of a Softmax actication with a shape of 3 instead of Sigmoid. The training data consists of approximately 386k samples for class 0, 2795 samples for class 1 and 464 samples for class 2. The main reason for this extension was that the binary classification model was missing out on critical information, as it doesn't distinguish between a articles that have not been purchased after a box and an actual purchase. This has been broken down with the additional label and therefore, the model can provide a more nuanced and comprehensive understanding of the data.

The model has been trained over 10 epochs. Accuracy and loss seem to coincide with the binary classification problem. However, when inspecting precision and recall the model performs significantly worse. For class 1 (box but no purchase) the model achieved a precision of 2% and a recall of 38% and for class 2 (box and purchase) a precision of 1% and a recall of 12%. When breaking down the model to binary classification problem with a classification threshold of 0.25,

the model achieved a recall of 54% which is 7% less compared to the first model. The labels also serve as a weighting scheme. This means by calculating the sumproduct of the labels and the corresponding probability, a final probability can be derived (e.g., 0*p1 + 1*p2 + 2*p3). On average, the model predicted a probability of 0.11 for label 0, 0.29 for label 1 and 0.35 for label 2. This means that samples of label 2 would be the highest in the ranking on average. However, the difference between label 1 and 2 is small and the model struggles to differentiate between those two classes. There are many possible reasons why a multi-label classification model might perform worse than a binary classification model which will be discussed in the following section.

4.2.1 Accuracy and loss

Both plots below show the training/validation accuracy and loss of the multi-label classification problem. As already mentioned, when comparing the accuracy to the binary classification problem not much difference is visible. The previously mentioned variance in validation loss and accuracy decreased.

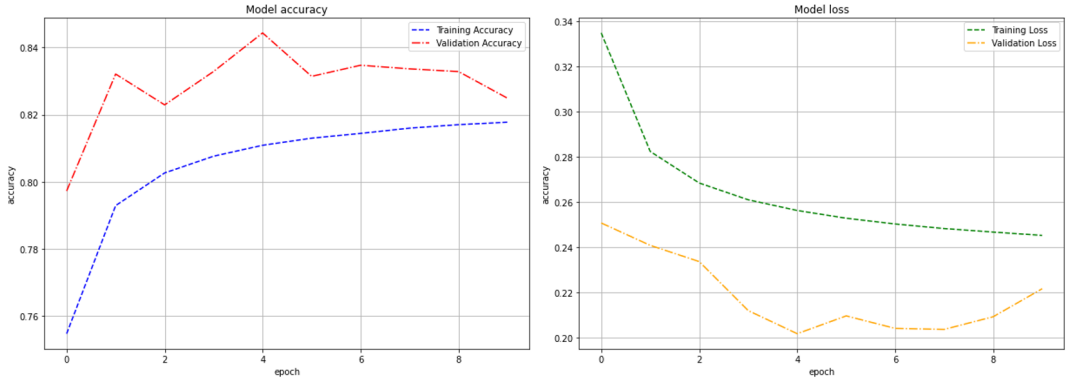

Figure 14: Accuracy & loss multi-label model

4.2.2 Precision, Recall, Confusion matrix and AUC-ROC

The table below shows class-specific results for the multi-label classification algorithm. It is observable that recall and precision decreased by a significant amount for both positive classes.

	Precision	Recall	F1-Score	Support
0	0.99	0.83	0.90	96,549
1	0.02	0.38	0.04	721
2	0.01	0.12	0.02	127

Figure 15: Precision & Recall of multi-label model

Figure 16 below shows the ROC curve and the confusion matrix of the multi-label classification problem after breaking it down to a binary classification problem. It is important to note that again a classification threshold of 0.25 has been used. The area under the ROC curve is slightly less compared to the first model. Additionally, model 1 identified 54 more true positives than model 2 resulting in a difference of 7% in recall.

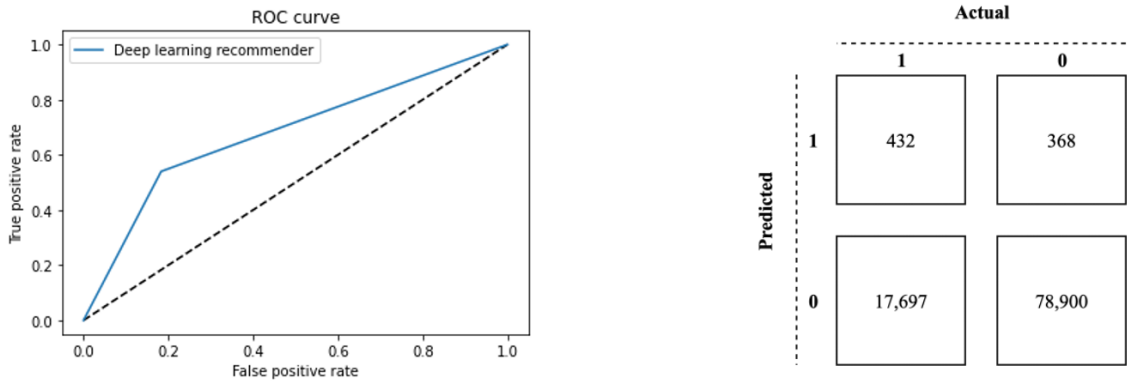

Figure 16: AUC-ROC & Confusion matrix of multi-label model

There are several possible reasons why extending a binary classification model to a multi-label classification problem could result in a decrease in performance for the positive classes. The most reasonable factors are listed below:

1. The additional labels may be more difficult to predict accurately. In the multi-label classification problem samples for the positive classes became significantly fewer as they were split up, which can make the task more complex and difficult to predict accurately (M.R. Boutell, 2004).

2. Labels may be correlated with one another. In a multi-label classification problem, it is common for different labels to be correlated with one another (Tsoumakas, 2010). If this is the case, the model may struggle to predict one label accurately if it is strongly correlated with another label that is difficult to predict, leading to a decrease in precision and recall.

	0	1	2
0	1	-0.926	-0.375
1	-0.926	1	-0.003
2	-0.375	-0.003	1

Figure 17: Label correlation

The table above shows a correlation matrix between all labels. Label 0 (no purchase) has a very strong inverse relationship with a correlation coefficient of -0.926. This could potentially justify why the model performs so well when predicting class 0. A high negative correlation indicates that when one label is present, it is very likely that the other label is not present. This could lead to confusion for the model when trying to predict both labels, as it may struggle to disambiguate between the two.

3. The additional labels are even rarer and have a skewed distribution. In these cases, it is difficult for the model to accurately predict them due to a lack of sufficient examples or a bias in the training data (Gama, 2014). The train set only offers 464 actual purchases with label 2, which can be considered very few compared to the 386.327 samples of label 0 (not chosen). It is questionable, even after oversampling, if this number is a sufficient feature representation of a purchase. Even samples of class 1 (box) become fewer when extending the model to a multi-label classification problem. Therefore, the model struggles to learn to characterize both positive classes and recall and precision suffer.

4. The model may need to be sufficiently fine-tuned. Simply adapting a binary classification model to a multi-label classification problem without any further fine-tuning may not be sufficient to optimize the model for the new task (Zhang, 2018).

In conclusion, various factors could contribute to a decrease in precision and recall when extending a binary classification model to a multi-label classification problem. Limitations and further improvements will be discussed in section 5.

4.3 Choice of model

In general, the choice between a binary classification model and a multi-label classification model will depend on the specific task at hand and the amount of data available. If the task requires predicting only a single label and there is a sufficient amount of data, a binary classification model may be the better choice due to its higher performance. However, if the task requires predicting multiple labels or there is a limited amount of data, a multi-label classification model may be a better choice due to its ability to consider more information. The TBYB approach of CurveCatch requires more than one label which means that in the long-term the choice should always fall on the multi-label classification problem.

Moreover, both models could also work together. If both models assigned a high probability of purchase to specific product it is more likely that the item will be purchased. Hence, it would make sense to use both models for the time being until the multi-label classification model is able to equal or even outperform the binary classification model.

05 DISCUSSION

This section focuses on the overall applicability of the model as well as its strengths, limitations, and further improvements in relation to the problem it was designed to address, which is predicting to what likelihood or probability a customer is going to buy a product.

5.1 Assessment of applicability

The goal of the work was to build a machine learning model that is able to partly replace the style expert in the process of making product recommendations. It is evident that the model would miss out on a high number of sales due to its inability to correctly identify purchases in the test set, which is reflected in the low recall and precision of the model. The question of whether CurveCatch can solely rely on the model in the future can be clearly answered in the negative. However, it is important to keep in mind that this work never aimed to fully replace the style expert. Rather, the goal was to build a machine learning model to supplement and support the human-in-the-loop component. Ultimately, the machine learning model provides a probability of purchase for each product in the inventory, serving as a starting point for the human expert to consider. However, due to the low confidence in the model scores, the human expert must also consider additional factors that may impact the likelihood of a purchase. These may include the customer's personal style and preferences, as well as any current trends or popular products in the market. By combining the insights from the machine learning model with the expertise of the style expert, it is possible to make more informed and accurate product recommendations. All things considered, the personalized deep learning recommender for CurvCatch can be a valuable source to boost efficiency of current product recommendations of the start-up. The style expert can always access the ranking of the products as a starting point but should always keep in mind that a major part of products the customer might buy were not identified by the model. However, the positive classes are still ranked higher than negative classes which underlines the value of the produced ranked list of items. Additionally, the model will gain even more relevance once the start-up scales up and grows, as the main reason for low confidence in predictions is the low number of purchases available in the training data. In the future is extremely crucial for CurveCatch to try to improve the recall of the model. For the style expert it takes a lot more time to look for products that still could be recommended with a low probability of purchase, than simply filtering out products the expert believes the customer won’t buy even when the model assigned a high probability of

purchase to the product. To put it in a nutshell, the model represents a starting point for the current scale of the start-up but needs to be extended once the start-up grows.

5.2 Strengths

A major strength of the deep learning model is the extensive feature representation for both users and products and that the model adequately leverages all implicit and explicit information available to CurveCatch. By one-hot encoding all relevant categorical features of the questionnaire and of the historical orders data approximately 333 features are generated that could potentially characterize a purchase between a specific user-product pair. While conventional recommendation systems, like collaborative filtering would completely miss out on explicit feedback, the personalized deep learning model can process a wide-ranging set of features. This means, the personalized deep learning recommender of CurveCatch can learn complex patterns in data and make more accurate recommendations compared to collaborative filtering algorithms, which rely on simple statistical models. Additionally, the model performs relatively well with sparse data and will perform even better once the start-up gathers more training data. For traditional approaches it is a challenge get confident results with sparse data and often many purchases are necessary to get accurate predictions. Another major advantage of the current architecture is that the model is able to handle new and cold items (items with no or few interactions). Collaborative filtering approaches heavily rely on item popularity and these models are prone to only recommend items that have been purchases multiple times while deep learning models predict a probability dynamically for each user-product pair. Furthermore, user preferences change over time. Especially in the fashion industry new trends come up quick. Deep learning models can adapt to changing user preferences and product availability in real-time, which is not possible with collaborative filtering algorithms that are based on static models. Lastly, deep learning models can handle multiple types of data which represents additional opportunities to further improve the model's performance. It is possible incorporate multiple types of data, such as text, images, and ratings, which can improve the quality of recommendations. While the incorporation of images remained out of scope for this work, it could be a important improvement for CurveCatch. More on this topic will be discussed in the next section. All in all, the model offers a variety of strengths compared to traditional approaches to build a recommender system. The model is specifically tailored to the data available to CurveCatch an also challenges crucial challenges CurveCatch is facing, e.g., data sparsity. Lastly, the model offers plenty of room for further improvements.

5.3 Limitations

One major disadvantage of neural networks, particularly deep learning models, is their lack of interpretability, often referred to as the "black box" problem. This means that it can be challenging to understand how the model is making decisions and to identify the specific features or input data that are driving its predictions. While some techniques such as saliency maps and layer-wise relevance propagation have been proposed to try to shed light on the internal workings of neural networks (Bach, 2015; Montavon, 2018), these methods are not always reliable and may not provide a complete understanding of the model's decision-making process. The lack of interpretability can make it difficult to trust the predictions made by the model and to identify and fix any errors or biases that may be present in the model's output (Doshi-Velez, 2017). For CurveCatch it would be very valuable to know which user features characterize purchases for a specific product.

One strength outlined in the previous chapter was the vast number of user & product features available to CurveCatch. On the other hand, the low number of sales represents a major limitation of this work which also makes it harder for the model to correctly predict the positive classes. Even after labeling all products per customer and synthetically generating new samples of the minority classes, the model struggles to characterize purchases which is reflected in a low recall. Overall, several actions have been taken to address data sparsity, however, it remains a major challenge and limitation of this work and the model’s predictions would get more accurate with more purchases.

5.4 Improvements

Considering that the multi-label classification problem only identified 12% of all purchases in the test set it is evident that the model needs further adjustments to be a reliable top-line recommendation system.

5.4.1 Generate more training data

The data offers a variety of meaningful user and product features but simply lacks purchases. This means the model is not able to accurately identify all of the instances of class 1 & 2 in the data, resulting in missed opportunities for sales. To improve the performance of the model and make it a more reliable source of predictions, several steps can be taken. One approach is to increase the amount of data available for training the model. This can help the model learn more about the

characteristics of purchases, allowing it to better identify them in the future. CurveCatch could extend the time frame of the historical order data to have access to more purchases. Furthermore, adding recent purchases, for example monthly, would boost the model's performance with not much additional effort.

5.4.2 Incorporation of domain knowledge

Deep learning models can often be improved through the incorporation of domain knowledge, such as heuristics or other expert-derived information. One way to incorporate domain knowledge into a neural network is through the use of heuristics. Heuristics are simple, rule-based strategies that can be used to make decisions or solve problems quickly and efficiently. They are often used when exact solutions are not feasible or when there is not enough time or resources to explore all possible options (Steven Walczaka, 1999). For CurveCatch the extensive knowledge of style experts and other employees could address data sparsity to improve the model's performance. A domain expert in the field of lingerie might have knowledge of certain heuristics that are associated with an increased probability of purchase for certain product. For example, certain user information gathered through the questionnaire could point to a certain category of bras. This information could be inputted to the model as additional features, along with the standard input features. Additional heuristical features could be represented as a binary variable which takes the value of 1 if the data already points to a specific category of bras. Another approach is to use heuristics to guide the learning process of the neural network. The domain expert might have knowledge of certain heuristics that are particularly relevant to the task at hand, such as the importance of certain factors that contribute to a purchase or the relationships between different features. By incorporating this knowledge into the training process, the neural network can potentially learn more effectively and achieve better performance. There are also more sophisticated approaches for incorporating heuristics into neural networks. One example is the use of expert systems, which are systems that are designed to mimic the decision-making processes of a human expert in a particular domain. Expert systems can be used to guide the learning process of a neural network, providing additional context and information that can help the model make more accurate predictions (C. Lacave, 2004). Overall, the incorporation of domain knowledge, such as heuristics, can be a powerful way to improve the performance of a neural network. By leveraging expert-derived information, the model can make more informed decisions and better adapt to new situations, resulting in improved accuracy and reliability.

5.4.3 Extension through visual images

In addition to the data already described, CurveCatch also has access to images of most of the articles. One way that images can improve deep learning recommenders is by providing additional information about the items being recommended. For example, images of bras can provide visual cues about the style, color, and fit of the items, which may not be captured by text-based descriptions alone (He, 2016).

Another way how images can improve deep learning recommenders is by providing a more robust and diverse set of features for the model to learn from. When working with text-based data, the model may be limited by the vocabulary and syntax of the language being used. By contrast, images provide a rich source of visual features that can be extracted and used by the model, such as color, texture, and shape (Liu J. W., 2018). This can enable the model to learn more complex and nuanced patterns in the data, leading to improved recommendations.

Incorporating images into the recommendation process can improve the performance of deep learning models in several ways, including providing additional information about the items being recommended, enhancing user engagement, and providing a more diverse set of features for the model to learn from. Hence, many limitations and challenges CurveCatch must face in the future could potentially be tackled by the incorporation of images.

5.4.3 Extension to a ranking algorithm

Ranking algorithms, such as RankNet from Microsoft, have increasingly become the preferred approach for many recommendation systems due to their superior performance and ability to handle complex ranking tasks (Burges, 2010). Ranking algorithms can take into account the unique preferences of each user and the overall diversity of the item set, resulting in more personalized and diverse recommendations. The goal is to not just predict the probability of purchase or to predict whether the item belongs to a specific class, instead so-called learning-to-rank (LTR) algorithms predict the correct ordering of items.

A neural network can be extended to a learning-to-rank (LTR) algorithm by adding a ranking loss function to the network's objective. The ranking loss function measures the quality of the rankings

produced by the network and provides a measure of how well the network is learning to rank the items. There are several different ranking loss functions that can be used in an LTR neural network, including the pairwise loss and the listwise loss. The pairwise loss measures the difference between the scores of pairs of items, with the goal of correctly ranking the more relevant item higher. The listwise loss measures the difference between the predicted ranking and the ground truth ranking of the entire list (Cao, 2007; Liu, 2009). To train an LTR neural network, the network is fed a dataset of labeled examples, where the goal is to predict the relative order of the items in the list. The network is then optimized using the ranking loss function as the objective, with the aim of minimizing the loss and producing accurate rankings (Wang, 2017).

A reliably learned ranking would save a lot of time of the style expert and would bring CurveCatch on steps closer to an automated multi-stage recommendation system. When customers want a box passed to their home, they first send a query by answering the questionnaire. The LTR algorithm would now automatically output a ranking of all items in respect to the query. The architecture of such a algorithm is significantly more complicated than a conventional classification task. Additionally, it is hard to develop a LTR algorithm with sparse data. For CurveCatch this could be interesting in the future as the current model architecture seems sufficient for the scale of the start-up.

06 CONCLUSION

In this work the task to optimize product recommendations of a Belgium fashion start-up called CurveCatch was studied. The start-up sells women's lingerie products and relies on a TBYB-approach, which means the customer does not actively shop on a website, rather, several items will be passed home to the client based on a questionnaire every customer is obliged to answer. The surveys offer a variety of user features, like the favorite brand & color and specific details about the position and firmness of a customer's breast. Additionally, several product features have been defined, consisting of brand, color, category, and other details of the item. Furthermore, two different personalized deep learning architectures were introduced to predict to what probability a customer is going to buy a product. The first model relied on a binary classification approach while the second model represents an extension of the first model as a multi-label classification problem. Ultimately, they predict the label:

$$\boldsymbol{Model\ (1)}\ \boldsymbol{y_{ui}} = \{\mathbf{1}: \boldsymbol{purchase\ or\ box}, \mathbf{0}: \boldsymbol{not\ chosen}\)$$

$$\boldsymbol{Model\ (2)}\ \boldsymbol{y_{ui}} = \{\mathbf{2}: \boldsymbol{box\ and\ purchase}, \mathbf{1}: \boldsymbol{box\ but\ no\ purchase}, \mathbf{0}: \boldsymbol{not\ chosen}\)$$

After labeling each product in the inventory per customer, the positive classes ended up in a severe class imbalance which has been tackled by an oversampling technique called SMOTE. The results demonstrated that the binary classification model performed significantly better. Both precision and recall were higher as the multi-label model struggled to correctly identify samples from both positive classes in the test set. However, the binary classification model misses out on significant implicit information accessible to CurveCatch, as the model doesn't distinguish between a purchase and a recommendation by a style expert that the customer didn't buy. Hence, in the long-term the decision should always fall on the multi-label classification model.The main goal of this work was to support the style experts in the process of making accurate product recommendations, which has been reached. However, with a low to medium recall in both models the style expert should always carefully consider that many interesting products have been assigned with a low probability of purchase. The output of the model serves as a starting point for the style expert but is not yet able to fully replace the human-in-the-loop component. Several ways to improve the model, like the incorporation of heuristics or images have been outlined which offer valid opportunities to build up on this work. All in all, this work builds a good fundament to further improve the product recommendations at CurveCatch. The final code and a deeper look at the Result can be found in the Appendix.

07 BIBLIOGRAPHY

Vaswani, A. e. (2017). *Attention is all you need - Advances in neural information processing systems* .

K. Hajjar, R. S. (2021). Attention Gets You the Right Size and Fit in Fashion. In *Recommender Systems in Fashion and Retail* (S. 77-98).

Nestler, A. (2021). *SizeFlags: Reducing Size and Fit Related Returns in Fashion E-Commerce.* Zalando SE.

Klein G, K. Y. (2017). *Opennmt: Open-source toolkit for neural machine translation.* ACL.

Anam Bhatti, H. A. (2020). *E-commerce trends during COVID-19 Pandemic.*

Hannu Saarijärvi, U.-M. S. (2017). *Uncovering consumers' returning behaviour: a study of fashion e-commerce.*

Choi, T.-M. (2016). *Analytical Modeling Research in Fashion Business.* Springer.

Ratcliff, C. (2014). *How fashion ecommerce retailers can reduce online returns.* Econsultancy Saatavissa.

A.R. Greenbaum, T. H. (2003). An investigation of the suitability of bra fit in women referred for reduction mammaplasty. British Journal of Plastic Surgery.

K. Wood, M. C. (2008). *Breast size, bra fit and thoracic pain in young women: a correlational study. Chiropractic and Osteopathy.*

Steele, E. M. (2010). *Optimising breast support in female patients through correct bra fit: a cross-sectional study.* Journal of Science and Medicine in Sport.

J. White, J. S. (2012). *Evaluation of professional bra fitting criteria for bra selection and fitting in the UK.*

Jaechoon Jo, S. L. (2020). *Development of Fashion Product Retrieval and Recommendations Model Based on Deep Learning.*

X. Chen, Y. Z. (2019). Dynamic Explainable Recommendation based on Neural Attentive Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 53-60.

G. Linden, B. S. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet Comput*, 76-80.

Pankaj Agarwal, S. V. (2018). *Personalizing Similar Product Recommendations in Fashion E-commerce.*

Justin Basilico, T. H. (2004). *Unifying Collaborative and Content-Based Filtering.* Brown University.

Pazzani, M. J. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review, 13*, 393 - 408.

Burke, R. (2007). Hybrid Web Recommender Systems . *The adaptive web*, 377-408.

Prem Melville, R. J. (2002). Content-Boosted Collaborative Filtering for Improved Recommendations. *AAAI-02 Proceedings*, 187-192.

Heng-Tze Cheng, L. K. (2016). *Wide & Deep Learning for Recommender Systems.* Google Inc.

Xiangnan He, L. L.-S. (2017). *Neural Collaborative Filtering∗.* Singapore.

Yehuda Koren, R. B. (2009). *Matric factorization techniques for recommender systems.* IEEE Computer Society.

F.O. Isinkaye, Y. F. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 216-273.

Long-Sheng Chen, F.-H. H.-C.-C. (2008). Developing recommender systems with the consideration of product profitability for sellers. *Information Sciences*(178), 032–1048.

Gawesh Jawaheer, M. S. (2010). Comparison of Implicit and Explicit Feedback from an Online Music Recommendation Service. *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec '10)*, 47–51.

Blerina Lika, K. K. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 2065-2073.

Andrew I. Schein, A. P. (2002). Methods and Metrics for Cold-Start Recommendations. *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 253 - 260.

Khoshgoftaar, J. M. (2019). *Survey on deep learning with class imbalance.* Journal of Big Data.

Nitesh V. Chawla, K. W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research 16*, 321–357.

Max Kuhn, K. J. (2018). *Applied Predictive Modeling.* Springer.

Yang Zhang, F. F. (2021). Recommendation, Causal Intervention for Leveraging Popularity Bias in. *SIGIR '21: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 11–20 .

Stitchfix. (12. 12 2022). *Stitchfix: Algorithm tour*. Abgerufen am 09. 11 2022 von https://algorithms-tour.stitchfix.com/#recommendation-systems

O'Donovan J., L. Y. (2014). Flexible tensor factorizations for context-aware recommendation. *Proceedings of the 8th ACM Conference on Recommender Systems*, 133-140.

Y. Koren, R. B. (2009). Matrix factorization techniques for recommender systems. *Computer: Volume: 42, Issue: 8*, 30 - 37.

Shuai Zhang, L. Y. (2019). *Deep Learning Based Recommender System: A Survey and New Perspectives.* ACM Comput. Surv. 52, 1, Article 5.

Hollander, J. (2018). *The business of fashion: Designing, manufacturing, and marketing.* New York.

P. Kaur, J. S. (2020). Clothing size and shape: Determinants of clothing purchases among women. *International Journal of Consumer Studies*, 131 - 139.

Fry, C. (2017). *The psychology of fashion.* London.

K. Chung, J. K. (2021). The impact of body size and fashion involvement on fit evaluation and purchase intention. *Clothing and Textiles Research Journal*.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters 27(8)*, 861-874.

Flach, P. (2012). ROC curves and precision-recall curves: What are they and how do you use them? *Journal of Machine Learning Technologies*, 1-32.

J.A. Hanley, B. M. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology 143(1)*, 29-36.

Powers, D. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, 37-63.

Stehman, S. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment 62(1)*, 77-89.

Christopher D. Manning, P. R. (2008). *Introduction to Information Retrieval.*

Mark D. Smucker, J. C. (2007). *A Review of Evaluation Measures for Binary Classification Tasks.*

M.R. Boutell, J. L. (2004). Learning multiple labels in the presence of label correlations and label dependencies. *In Proceedings of the 21st international conference on Machine learning*, 259-266.

Gama, J. Z. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR), 46(4)*, 44.

Tsoumakas, G. K. (2010). Mining multi-label data. *In Data mining and knowledge discovery handbook*, 667-685.

Zhang, M. L. (2018). A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 283-296.

Burges, C. J. (2010). From RankNet to LambdaRank to LambdaMART: An Overview. *Microsoft Research Technical Report*.

Cao, Z. L. (2007). Learning to rank: from pairwise approach to listwise approach. *In Proceedings of the 24th international conference on Machine learning*, 129-136.

Liu, T. Y. (2009). Learning a ranking function using bias-enhanced metacognitive knowledge. *Journal of Machine Learning Research*, 961-999.

Wang, H. &. (2017). Learning to rank with non-smooth cost functions. *n Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 405 - 414.

Steven Walczaka, N. C. (1999). Heuristic principles for the design of artificial neural networks. *Information and Software Technology 41*, 107 - 117.

C. Lacave, C. D. (2004). A review of explanation methods for heuristic expert systems. *he Knowledge Engineering Review, 19(2)*, 133-146.

He, X. C. (2016). Clothing recommendation using deep learning on consumer reviews and images. *In Proceedings of the 25th International Conference on World Wide Web*, 693-703.

Liu, J. W. (2018). Deep learning for fashion recommendation: A survey. *ACM Transactions on Intelligent Systems and Technology*, 1-29.

Lipton, Z. (2016). The mythos of model interpretability.

Bach, S. B. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation.

Montavon, G. L. (2018). xplaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition, 72*, 11-22.

Doshi-Velez, F. &. (2017). Towards a rigorous science of interpretable machine learning.

Liu, Y. C. (2021). Set2setRank: Collaborative Set to Set Ranking for Implicit Feedback based Recommendation. *Lei Chen, Le Wu , Kun Zhang1,2 , Richang Hong1,2 , Meng Wang1,2,3*, 585-594.

Lei Chen, L. W. (2021). Set2setRank: Collaborative Set to Set Ranking for Implicit Feedback based Recommendation. *SIGIR '21*, 585-594.

Gawesh Jawaheer, P. W. (2014). Modeling User Preferences in Recommender Systems: A Classification Framework for Explicit and Implicit User Feedback. *ACM Transactions on Interactive Intelligent Systems, Vol. 4, No. 2,*, Article 8.

Demiriz, A. (2004). Enhancing Product Recommender Systems on Sparse Binary Data. *Data Mining and Knowledge Discovery, 9*, 147-170.

Deuk Hee Park, H. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications 39*, 10059-10072.

Alpaydin, E. (2014). Introduction to machine learning.

Hunt, C. A.-U. (2015). The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manage. Inf. Syst. 6, 4*, Article 13.

G. Linden, B. S. (2003). Amazon.com recommendations: item-to-item collaborative filtering . *EEE Internet Computing, vol. 7, no. 1*, 76-80.

Buckland, M. a. (1994). The relationship between Recall and Precision. *J. Am. Soc. Inf. Sci., 45*, 12-19.

Christopher D. Manning, P. R. (2009). *An Introduction to Information Retrieval.*

08 TABLE OF FIGURES

09 APPENDIX

Link to GitHub Repository:

https://github.com/paul-ruh/CurveCatch