European Journal of Operational Research 279 (2019) 620–634

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Evaluating multi-label classifiers and recommender systems in the financial service sector

Matthias Bogaert[a,b,*], Justine Lootens[b], Dirk Van den Poel[b], Michel Ballings[c]

[a] The University of Edinburgh Business School, 29 Buccleuch Place, Edinburgh EH8 9JS, UK
[b] Department of Marketing, Ghent University, Tweekerkenstraat 2, 9000 Ghent, Belgium
[c] Department of Business Analytics and Statistics, The University of Tennessee, 916 Volunteer Blvd., 249 Stokely Management Center, 37996 Knoxville, TN, USA

ARTICLE INFO

Article history:
Received 20 September 2017
Accepted 29 May 2019
Available online 6 June 2019

Keywords:
OR in marketing
CRM
Predictive modeling
Multi-label classifiers
Recommender systems

ABSTRACT

The objective of this paper is to evaluate multi-label classification techniques and recommender systems for cross-sell purposes in the financial services sector. We carried out three analyses using data obtained from an international financial services provider. First, we tested four multi-label classification techniques, of which the two problem transformation methods were combined with several base classifiers. Second, we benchmarked the performance of five state-of-the-art recommender approaches. Third, we compared the best performing multi-label classification and recommender approaches with each other. The results identify user-based collaborative filtering as the top performing recommender system, with a cross-validated F_1 measure of 42.20% and G-mean of 42.64%. Classifier chains binary relevance with adaboost and binary relevance with random forest are the top performing multi-label classification algorithms for respectively F_1 measure and G-mean, yielding a cross-validated F_1 measure of 53.33% and G-mean of 54.37%. The statistical comparison between the best performing approaches confirms the superiority of multi-label classification techniques. Our study provides important recommendations for financial services providers, who are interested in the most effective methods to determine cross-sell opportunities. In previous studies, multi-label classification techniques and recommender systems were always investigated independently of each other. To the best of our knowledge, our study is therefore the first to compare both techniques in the financial services sector.

1. Introduction

Cross-selling is omnipresent in today's business reality. It is common knowledge that acquiring new customers is much more expensive than increasing sales by cross-selling to existing customers (Reinartz & Kumar, 2003). In order to perform cross-sell analysis, two approaches can be distinguished in literature, namely statistical models and recommender systems. Cross-sell analysis based on statistical models is mostly conducted with multi-label classification algorithms. These algorithms are able to define which products a customer will buy next and are widely used in a variety of application domains. Recommender systems can also be used in the context of cross-sell analysis (Resnick, Iacovou, Suchak, Bergstrom, & Riedl, 1994; Shardanand & Maes, 1995). As mentioned by Lü et al. (2012), recommender systems are able to transform user preferences into predictions of their likes and interests. Since the rise of recommender systems, research into statistical models for cross-sell purposes has faded into the background.

Both statistical models and recommender systems have been extensively studied with regard to cross-sell. However, these techniques were always investigated independently of each other. For this reason, a comparison between statistical models and recommender systems could be useful to discover the most effective technique to deal with cross-sell analysis. This comparison was already identified as an issue to be researched by Knott, Hayes, and Neslin (2002), and knowing the most effective cross-sell modeling approach would be beneficial for companies, as cross-selling plays an important role in improving customer retention and thus has a significant impact on the sales perspectives and viability of companies (Kamakura, Wedel, de Rosa, & Mazzon, 2003).

To address this gap in literature, we perform three analyses using data of an international financial services provider. First, we test several multi-label classification techniques, following the

* Corresponding author at: University of Edinburgh Business School, 29 Buccleuch Place, Edinburgh EH8 9JS, UK.
E-mail addresses: matthias.bogaert@ed.ac.uk (M. Bogaert), justine.lootens@ugent.be (J. Lootens), dirk.vandenpoel@ugent.be (D. Van den Poel), michel.ballings@utk.edu (M. Ballings).

https://doi.org/10.1016/j.ejor.2019.05.037
0377-2217/

Table 1
Overview of recent cross-sell literature in the financial services sector. RS = Recommender Systems, ML = Multi-Label classification.

Studies	ML	RS
Kamakura, Ramaswami, and Srivastava (1991)	X	
Kamakura et al. (2003)	X	
Li, Sun, and Wilcox (2005)	X	
Liu and Cai (2007)	X	
Li, Sun, and Montgomery (2011)	X	
Felfernig and Kiener (2005)		X
Felfernig, Isak et al. (2007), Felfernig, Teppan, and Gula (2007)		X
Gonzalez-Carrasco, Colomo-Palacios, Luis Lopez-Cuadrado, Garcia-Crespo, and Ruiz-Mezcua (2012)		X
Abbas, Bilal, Zhang, and Khan (2015)		X
Musto, Semeraro, Lops, de Gemmis, and Lekkas (2015)		X
This study	**X**	**X**

recommendation of Madjarov, Kocev, Gjorgjevikj, and Dzeroski (2012). Second, we benchmark multiple recommender systems based on the framework of Geuens, Coussement, and De Bock (2018) and assess their applicability in the financial services sector. Third, we compare the best recommender system and multi-label classification techniques in terms of performance and usability. Consequently, we identify the best approach to perform cross-sell analysis in the financial services sector.

The remainder of this paper is organized as follows. In the next section, we give an overview of cross-sell literature in the financial services sector, which clearly indicates the lack of studies comparing statistical models and recommender systems. Next, we describe our empirical comparison study, including the data, variables, analytical techniques, model evaluation criteria and cross-validation. We then elaborate on the results and conclude this study. In the penultimate section, we discuss practical implications. Finally, we consider the limitations and directions for future research.

2. Literature review

An overview of cross-sell literature would be too extensive, because of the plentitude of literature available. Since our data were provided by an international financial services provider, we focus on recent literature about cross-sell in the financial services sector, as can be seen in Table 1. Cross-selling literature can be classified based on the used methodology, namely multi-label classification or recommender system. We note that cross-sell applications can also be solved with other statistical models; binary and multi-class classification. The former class of algorithms focuses on one specific product and tries to determine the most appropriate customers to sell this particular product to (Thuring, Nielsen, Guillen, & Bolance, 2012). The latter class of algorithms considers the whole range of products and selects out of this range the product that is most likely to be bought by a particular customer (Prinzie & Van den Poel, 2008). In cross-sell literature, this is often referred to as next-product-to-buy models when only one product is recommended (Knott et al., 2002). Since binary and multi-class algorithms cannot be compared with recommender systems approaches (i.e., their recommendation focuses on only one product), these studies are out-of-scope.

Recommender systems and multi-label approaches both try to come up with a set of relevant labels (or items) for each user. Recommender systems suggest the most appropriate products or services to particular users by predicting a user's preference for an item relying on information about the items, the users and the interactions between items and users (Lu, Wu, Mao, Wang, & Zhang, 2015). Multi-label classification algorithms classify instances (in our case one user to which we want to recommend financial products) into several non-overlapping classes (or labels) (Sokolova & Lapalme, 2009; Zufferey et al., 2015). Hence, the main purpose of a recommender system is to uncover the user's preferences and come up with personalized recommendations while the main goal of a multi-label classifier is to correctly assign a set of labels to each instance. Multi-label classification can be considered as a particular case of recommender systems in which user preferences are used as input to correctly assign items to each user. Table 1 shows the applications of multi-label algorithms and recommender systems for cross-selling in the financial services. Translated into a cross-sell context, multi-label classification algorithms consider the whole range of products and select out of this range multiple products that are most likely to be bought by a particular customer. An example of a multi-label classification problem can be found in Li et al. (2005). They present a dynamic multivariate probit model based on switching costs. Their model is able to define to whom, when and which products and services to cross-sell. The model assumes that decisions about purchasing different products are dependent upon each other. An application of recommender systems related to financial services can be found in Felfernig, Isak, Szabo, and Zachar (2007). Their recommender system follows three consecutive steps; (i) the definition of a recommender knowledge base including customer properties, product properties and constraints, (ii) the specification of the recommender process and (iii) the calculation of the recommendations. Their recommender system has been successfully applied to generating recommendations for investment products, financial services and financial support opportunities for students.

Table 1 clearly indicates that our study is the first to compare multi-label classification with recommender systems for cross-sell analysis in the financial services industry. Both multi-label classification techniques and recommender systems have been studied separately in this sector, but the two techniques have never been compared.

The contributions of this paper are threefold. First, we test several multi-label classification techniques, which can be categorized into problem transformation methods and algorithm adaptation methods (Tsoumakas & Katakis, 2007). Problem transformation methods adapt the data in such a way that binary classifiers can be used. Algorithms adaptation methods adapt the algorithm itself such that they can be applied to multi-label cases. The implemented multi-label classification techniques are multi-label random forest, multi-label random ferns, binary relevance and classifier chains. The latter two algorithms are problem transformation techniques and used in combination with several binary classifiers. Several authors (Debaere, Coussement, & De Ruyck, 2018; Read, Pfahringer, Holmes, & Frank, 2009) demonstrate the dominance of the classifier chains method over the binary relevance method. Moreover, Probst, Au, Casalicchio, Stachl, and Bischl (2017) prove that the classifier chains method consistently outperforms multi-label random forest and multi-label random ferns on multiple data sets. However, the efficiency of the binary relevance and classifier chains methods is mainly dependent on the choice of the binary classifier (Nair-Benrekia, Kuntz, & Meyer, 2015). The decision tree is a popular binary classifier for binary transformations in the multi-label setting and very appropriate in our case, since decision trees yield high performance in combination with large datasets containing a smaller number of features (Madjarov et al., 2012; Nair-Benrekia et al., 2015). Other techniques such as naive Bayes, random forest and stochastic boosting are also often used as binary classifiers in multi-label problems (Probst et al., 2017). Overall, the performance of multi-label classification techniques is mostly determined by the dataset and area of application (Nair-Benrekia et al., 2015). Therefore, we evaluate several multi-label classification techniques and several binary classifiers for the financial services sector.

A second contribution is that we benchmark several recommender systems: user-based collaborative filtering, item-based

collaborative filtering, recommender systems relying on association rules, the popular recommender method and the random recommender method. According to the experiment of Zhao and Cen (2013, pp. 117–151) on the binary Jester5k dataset, user-based collaborative filtering is the best performing recommender system approach, followed by recommender systems using association rules, item-based collaborative filtering, the popular recommender method and the random recommender approach. If the number of recommended items is small, the popular recommender method outperforms recommender systems relying on association rules and item-based collaborative filtering. Other studies (Breese, Heckerman, & Kadie, 1998) confirm the (modest) superiority of user-based collaborative filtering over item-based collaborative filtering, but stress the latter's scalability. These findings are partially confirmed by Geuens et al. (2018) who found that user-based collaborative filtering significantly outperformed item-based collaborative filtering for high sparsity levels (i.e., the percentage of products that are not bought). The opposite was found for low sparsity levels. Mobasher, Dai, Luo, and Nakagawa (2001), on the other hand, believe that recommender systems using association rules can outperform user-based collaborative filtering in some cases. In sum, most studies conclude that a recommender system's performance is highly dependent upon the application and the characteristics of the dataset (Herlocker, Konstan, Terveen, & Riedl, 2004). Hence, we contribute to literature by investigating the usability of recommender systems in the financial services sector.

The third contribution lies in comparing the best recommender system and multi-label classification techniques to determine the best overall approach in identifying cross-sell opportunities. Both techniques have their strengths: recommender systems are good at dealing with information overload (Geuens et al., 2018), whereas multi-label classification techniques are able to take into account drivers of human choice behavior (Zhang & Zhou, 2014). Nevertheless, the comparison between multi-label classification techniques and recommender systems is far under-researched. To fill this gap in literature, we compare multiple multi-label classification techniques and recommender systems for the financial services sector.

3. Methodology

3.1. Data

Our data were provided by an international financial services provider. The dataset represents its Belgian customer base on December 1, 2016, containing 2,923,989 unique customers and 6808 unique products. Each customer occurs in as many observations as he/she owns products. In total, there are 6,065,612 observations, meaning that the 2,923,989 customers possess 6,065,612 products. As the financial services provider has both a bank and insurance department, the data contain purchase history and customer characteristics related to both departments.

To solve any data quality issues, we performed the following data pre-processing steps. We only selected non-deceased individual customers. For these selected customers, we only included their products with an active product status, meaning that the product purchase passed through all administrative compliances and was not delayed, modified, cancelled or liquidated. Moreover, products with high sparsity levels (i.e., products having a sparsity level smaller than 0.25%) were removed from the dataset. After removal of the sparse items, the number of products for analysis decreased from 6808 to 83. We experimented with several sparsity levels and chose the level that allowed a maximum decrease in sparsity while maintaining as much information as possible. The removal of sparse products was performed to avoid problems during cross-validation, since several classification approaches cannot deal with constant or extremely sparse labels (Probst et al., 2017).

Table 2
Summary statistics of number of distinct products a customer has purchased.

Distinct number of products required for customers to be included in analysis	Min	1st Qu.	Median	Mean	3rd Qu.	Max.
No restriction	1.00	1.00	1.00	1.663	2.00	16.00
≥ 3	3.00	3.00	3.00	3.992	5.00	16.00
≥ 5	5.00	5.00	6.00	5.93	6.00	16.00

Finally, we also omitted the customers owning products with high sparsity levels from the analysis. The final dataset comprises 2,074,748 customers, possessing 3,898,831 products.

Subsequently, we only selected customers having five or more distinct products, leaving us with 96,696 customers. This condition was imposed because of the specific evaluation procedure of a recommender system: a recommender system's prediction phase consists of generating recommendations for customers of whom a certain number of randomly picked products (i.e., g given items) are provided to the recommender system to base its recommendations on. The actual performance evaluation then consists of determining whether the recommended products correspond to the remaining products owned by these customers (i.e., all owned products minus the g given items). If, however, we take into account all 2,074,748 customers in the dataset**,** Table 2 indicates that 75% of these customers own two or less distinct products (below or equal to the third quartile). If in that case two products are provided to the recommender system for evaluation (i.e., $g=2$), all possible additional products that the recommender system recommends will be considered as incorrect for 75% of the customers. Hence, it is recommended to set the minimum of the distinct number of products larger than the number of items given (g). By doing so the recommended products can be compared to the products that the customers possess. The number of items given (g) range from 2 to 5 (Breese et al., 1998). Hence, we decided to only select customers with a minimum of 5 distinct products (Table 2). For example, if we include customers with 3 or more products we see that 75% of the observations is between 3 and 5 (Table 2), if 5 items are then given to the recommender system any additional product will also be considered as incorrect. Therefore, the minimum should be set to 5 such that the recommender systems can come up with good and relevant recommendations. The fact that only 5% of the initially selected customers satisfies the later imposed selection criterion is not problematic since the sample is still large enough to come up with reliable recommendations. Moreover, from an application perspective, it is more effective to fit models and evaluate their performance on smaller samples but with customers possessing a sufficient number of products. Afterwards these models can be reliably applied on the complete customer base (Prinzie & Van den Poel, 2008). As a final data cleaning step, we deleted customers with a negative age (i.e., users who were born after the data collections), negative recency (i.e., users who bought financial products after the data collection) and negative length of relationship (i.e., users who became a client after the data collection). The final data set then comprised 96,602 unique customers.

3.2. Variables

We translated all available information into 38 predictors, which are used by the multi-label classification techniques in this study. Appendix A provides an overview of these 38 predictor variables. The variables can be classified in the following three categories: non-behavioral and non-company specific variables, non-behavioral and company-specific variables, and behavioral and company-specific variables (Van den Poel, 2003). Owner_Ind

Table 3
Summary statistics of distribution of total number of distinct financial products owned by all customers.

Min	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0013	0.0083	0.0236	0.0715	0.0755	0.5773

is an example of a non-behavioral, non-company specific variable and indicates whether a customer is owner of a house/apartment on December 1, 2016. Employee_Ind is an example of a non-behavioral, company-specific variable and indicates whether the customer is an employee of the company. Frequency can be categorized as a behavioral and company-specific variable and represents the number of products the customer possesses on December 1, 2016.

In total, there are 83 binary dependent variables, representing the customers' ownership of financial products. Such binary variables are equal to 1 if a customer owns a specific product and to 0 if not. Table 3 shows the summary statistics of the distributions of the 83 binary dependent variables in our sample. In our case, the distribution of a binary dependent variable represents the proportion of ones of a certain financial product (i.e., the proportion of customers in our sample that possess a certain financial product). For example, the most popular financial product in our sample is possessed by 57.73% of all customers.

3.3. Analytical techniques

In this section, we elaborate on the different analytical techniques used to determine cross-sell opportunities. These techniques can be classified into multi-label classification techniques and recommender systems. Table 4 contains a tabular overview of the advantages, disadvantages and the expected performance of each technique in this study.

3.3.1. Multi-label classification techniques

According to the extensive study of Madjarov et al. (2012) and Debaere et al. (2018), multi-label classification techniques can be divided into problem transformation methods and algorithm adaptation methods. We note that ensemble methods are seen as a part of algorithm adaptation methods, since we include ensembles of algorithm adaptation methods. All these methods are implemented using the *R*-package *mlr* (Bischl et al., 2016).

Problem transformation methods. Problem transformation methods restructure the multi-label problem into binary classification,

Table 4
Summary of multi-label algorithms and recommender systems.

Multi-label algorithms			
Technique	Pros	Cons	Performance
Problem transformation method			
Binary relevance	-Low computational overhead -Simple and intuitive -Can be parallelized -Robust to overfitting	-No label dependencies -Sensitive to class imbalance	-Reasonable performance
Classifier chains	-Accounts for label dependencies	-Slow -No parallelization -Sensitive to label ordering	-Better than binary relevance.
Decision trees	-Easy to understand -Fast	-Prone to overfitting -Unstable	-Average
Naive Bayes	-Easy and intuitive -Low computation time	-Assumptions often violated	-Below average
Neural networks	-Handles complexity -Robust to overfitting	-Large computation overhead -Blackbox	-Above average
Random ferns	-Prone to overfitting -Can be parallelized	-Tuning required -Lack of interpretation	-Above average
Adaboost	-Best off-the-shelf classifier -Few parameters	-Large computational overhead -No parallelization -Prone to overfitting	-Top performer
Random forest	-Prone to overfitting -Few parameters -Can be parallelized	-Can perform poorly with unbalanced classes	-Top performer
Algorithm adaptation methods	-Simple extension of binary algorithms -Original data input	-Not all algorithms can be adapted to multi label setting	-Better than problem transformation methods
Random ferns	-Prone to overfitting -Few parameters -Can be parallelized	-Can perform poorly with unbalanced classes	-Top performer
RF	-Prone to overfitting -Few parameters -Can be parallelized	-Can perform poorly with unbalanced classes	-Top performer
Recommender systems			
UBCF	-Better with a lot of items	-Cold start problem -Scalability (computational overhead)	-Above average
IBCF	-More memory efficient -Better with a lot of users	-Cold start problem -Sensitive to high sparsity	-Top performer
AR	-Deals well with sparsity -Fast -Takes into account high order effects	-Very sensitive to parameter settings -Too many rules	-Average
POPULAR	-Easy and intuitive -Fast -Reasonable performance	-Often too easy	-Average

multi-class classification or label ranking problems (Zhang & Zhou, 2014). In other words, problem transformation methods adapt the input data in such a way that the algorithms can be used without adjustments. This means that only the input data are transformed and the algorithms remain unchanged. In this study, we focus on binary relevance multi-label algorithms for our problem transformation methods (Bischl et al., 2016). The reason for this focus is that binary relevance problems resemble recommender systems the most. In binary relevance multi-label problems are transformed in one-versus-all binary classification problems, which is the same as the binary user-item matrix in recommender systems. Hence, the input of the binary relevance approaches and the recommender systems are the same in this case. The two most well-known binary relevance transformation methods are binary relevance and classifier chains (Madjarov et al., 2012).

The binary relevance method constructs a binary classifier for each label and combines the binary predictions of these classifiers (Probst et al., 2017). This method is simple, intuitive, can easily be parallelized and is robust to overfitting (Read et al., 2009; Zhang & Zhou, 2014). Further, the binary relevance method has a low computational complexity, proportional to the number of labels (Read et al., 2009; Zhang & Zhou, 2014). However, the binary relevance model does not take into account correlations among labels (Probst et al., 2017; Read et al., 2009; Zhang & Zhou, 2014).

On the contrary, classifier chains take into account label dependencies. This method also performs as much binary transformations as there are labels. Additionally, the feature space of each binary classification problem is extended with the true label information of all previous labels during the training phase (e.g., the feature space of the binary classification problem regarding the kth label consists of the 38 predictor variables and the $k-1$ previous labels) (Read et al., 2009). At prediction time, the true label information is replaced by the predicted labels (Probst et al., 2017) and as such, each label can only be predicted if its preceding labels in the chain have been predicted. The order of the chain is randomly chosen or follows a predefined sequence. The classifier chains method has an acceptable computational complexity, but cannot be parallelized in the prediction phase due to its iterative nature (Read et al., 2009; Zhang & Zhou, 2014). The performance of the classifier chains method is influenced by four factors: the length of the chain, the order of the chain, the dependency among labels, and the base classifiers (Read et al., 2009). In our case, the number of the labels was reduced from 6808 to 83, which should have a positive impact on the classifier chains' performance. Besides, we ordered the labels according to their number of occurrences in the data sample (from high to low) assuming that often purchased items are easier to predict than rarely purchased items. Finally, and given that the labels in this study represent financial products, we can assume that the dependency requirement is satisfied (Prinzie & Van den Poel, 2006).

Finally, we implemented the following base classifiers: decision trees (DT), naive Bayes (NB), neural networks (NN), random ferns (rFerns), adaboost (AB) and random forest (RF). These methods were chosen since they have proven to yield superior performance in cross-selling applications (Knott et al., 2002; Prinzie & Van den Poel, 2008). We refer the reader to Appendix B for a detailed description of the binary base classifiers.

Algorithm adaptation methods. Algorithm adaptation methods adapt popular learning algorithms so that they can be applied directly to the multi-label classification problem. In other words, algorithm adaptation methods fit the algorithm to the data (Zhang & Zhou, 2014). In this study, we implemented multi-label random ferns (Kursa & Wieczorkowkska, 2014) and multi-label random forest (Breiman, 2001) using the statistical *R*-packages *rFerns* (Kursa, 2014) and *randomForestSCR* (Ishwaran & Kogalur, 2019). For a detailed explanation of random forest and random ferns, we refer the reader to Appendix B.4 and B.6. As these methods are ensemble methods of algorithm adaptation approaches, these methods can also be classified as ensemble approach according to the framework of Madjarov et al. (2012).

Both multi-label random ferns and multi-label random forest follow the same procedures as their binary counterparts and are implemented using the same parameter settings. In multi-label problems, each node of each tree serves all classes (Kursa & Wieczorkowska, 2014). For multi-label random forest, the splitting rule, namely composite normalized Gini index splitting, is the only element that differs from the binary random forest classifier (Probst et al., 2017).

3.3.2. Recommender systems

According to the framework of Geuens et al. (2018), there are three design issues when implementing collaborative filtering methods for binary purchase data: the input data, the collaborative filtering configurations, and the evaluation. For the input data, we work with real-life binary purchase validation data, so we don't alter the input data characteristics. For the collaborative filtering configurations there are again three steps: data reduction, the collaborative filtering method and the similarity calculation. The data reduction step is not applicable in our case. Since we want to compare multi label techniques with recommender systems we alter the data as little as possible to make a fair comparison. The data reduction step would transform the input matrix to a reduced non-binary matrix, making it impossible to compare the results with multi-label classifiers (i.e., they always use the non-reduced binary input matrix). The collaborative filtering methods used are item-based and user-based collaborative filtering. As similarity calculation methods we used the Jaccard index since this is the only measure that can be used with binary purchase data. For the evaluation, we employed the most widely-used measures; precision, recall, accuracy, F_1 and G-mean (see Section 3.4). Besides the collaborative filtering method, we also implemented recommender systems using association rules, the popular recommender method and the random recommender method (Hahsler, 2011). For each recommender system approach, we cross-validated the parameters given g, the number of randomly chosen items that are provided to the recommender algorithm for evaluation (Breese et al., 1998), and n, the number of recommendations to generate in the top-N lists. g ranges from 2 to 5 and n ranges from 2 till 8. We used the statistical *R*-package *recommenderlab* to implement the recommender systems (Hahsler & Vereet, 2019).

Collaborative filtering. Collaborative filtering is one of the most popular recommendation techniques and can be subdivided in user-based and item-based collaborative filtering (Lu et al., 2015). The former collaborative filtering technique recommends items to a particular user that are favored by similar users, while the latter recommends items similar to this user's previously preferred items. The similarity between users or items is derived from the user-item matrix, which contains for each user-item pair (i.e., each customer-product pair) a 0 or 1 representing the user's purchasing behavior (Leskovec, Rajaraman, & Ullman, 2014, chap. 9). The zeros in a binary user-item matrix can be interpreted in two ways: either the user does not like or need the item (i.e., negative preference) or the user does not know the item (i.e., unknown preference) (Hahsler, 2011). To circumvent the decision whether the zeros are negative examples or unknown, we make use of the Jaccard index as similarity measure. The Jaccard index, as can be seen in Eq. (1), only concentrates on matching ones (Hahsler, 2011). In Eq. (1), X and Y are the sets of items with a 1 in two particular user profiles (i.e., in case of user-based collaborative filtering) or the sets of users with a 1 in two particular item profiles (i.e., in

case of item-based collaborative filtering):

$$\text{sim}_{Jaccard}(X,\ Y) = \frac{|X \cap Y|}{|X \cup Y|}. \quad (1)$$

In user-based collaborative filtering, the Jaccard index is used to calculate pairwise similarities between users. In order to define a top-N list for a certain user, the *nn* most similar users (i.e., nearest neighbors) are selected. Then, the top-N list is composed based on the weighted averaged purchase behavior of these similar users, while excluding the items the user already possesses (Hahsler, 2011). As a result, it is important to define the optimal value for *nn*. Hence, we cross-validated the parameter *nn* by sequencing over all values of $nn = \{25, 50, 75, 100\}$. A disadvantage of user-based collaborative filtering is related to its scalability: the whole user-item matrix must be stored in memory and heavy similarity calculations must be performed in order to generate relevant recommendations (Hahsler, 2011).

In item-based collaborative filtering, the Jaccard index is used to calculate pairwise similarities between items. Item-based collaborative filtering is a model-based approach in which the model size can be reduced by only storing the k most similar items for each item. In order to define a top-N list for a certain user, the k most similar items first have to be determined for every item. Then, a score can be calculated for every item by taking into account the item similarities and the user's purchasing behavior. The n items with the highest scores, except the items the user already owns, are recommended (Hahsler, 2011; Leskovec et al., 2014, chap. 9). Consequently, it is important to define the optimal value for k. Hence, we cross-validated the parameter k by sequencing over all values of $k = \{10, 20, 30, 40, 50\}$. Item-based collaborative filtering is more efficient than user-based collaborative filtering, since item-based similarities are more static, which makes that the similarity matrix can be fully precomputed (Sarwar, Karypis, Konstan, & Riedl, 2001).

Both user- and item-based collaborative filtering suffer from the cold-start problem (Schein, Popescul, Ungar, & Pennock, 2002) and experience difficulties to deal with a sparse user-item matrix (Adomavicius & Tuzhilin, 2005). However, in this study, we alleviated both problems by only including customers having 5 or more products and focusing on the 83 most frequently bought products.

Recommender systems using association rules. According to Demiriz (2004), recommender systems using association rules are another way to deal with sparse binary data. These systems generate recommendations relying on a dependency model based on a set of association rules. Consider a user-item matrix containing a set of users $U = \{u_1, u_2, \ldots\ldots, u_m\}$ and a set of items $I = \{i_1, i_2, \ldots\ldots, i_n\}$. Association rules are derived from such a binary user-item matrix, in which each user is considered as a transaction $T \subseteq I$, containing the user's purchased items. An association rule is of the form $X \rightarrow Y$ where $X,\ Y \subseteq I$ and $X \cap Y = \emptyset$.

In order to limit the number of association rules to the most relevant ones, support and confidence are used as measures of significance and interestingness (Sarwar, Karypis, Konstan, & Riedl, 2000):

$$\begin{aligned}\text{support}(X \rightarrow Y) &= \text{support}\ (X \cup Y) \\ &= \frac{\text{number of transactions containing } X \cup Y}{\text{total number of transactions}}\end{aligned} \quad (2)$$

$$\begin{aligned}\text{confidence}(X \rightarrow Y) &= \frac{\text{support}(X \cup Y)}{\text{support}(X)} \\ &= \frac{\text{number of transactions containing } X \cup Y}{\text{number of transactions containing } X}\end{aligned} \quad (3)$$

Support (Eq. (2)) denotes the proportion of transactions containing both X and Y, whereas confidence (Eq. (3)) indicates the proportion of transactions containing Y, given that the transactions contain X. Only the rules satisfying $\text{support}(X \rightarrow Y) > s$, $\text{confidence}(X \rightarrow Y) > c$ and $|X \cup Y| \leq l$ are included in the dependency model (Hahsler, 2011). The last constraint (i.e., $|X \cup Y| \leq l$) limits the number of items included in an association rule; in this study l was set to 10 according to the recommendation of Hahsler et al. (2019). In combination with a low minimum support higher values of l lead to overfitting, whereas low value of l (e.g., 2 and 4) in combination with low minimum support and confidence may give infeasible solutions. Moreover, the performance of association rules is mainly driven by the support and confidence parameter (Lin, Alvarez, & Ruiz, 2002). We cross-validated the parameters s and c by sequencing over all combinations of values of $s = \{0.01, 0.05, 0.1\}$ and $c = \{0.6, 0.7, 0.8, 0.9\}$, following the example of Demiriz (2004) and Lin et al. (2002). The lower s and c and the higher l, the bigger the model size and consequently the longer the computation time (Hahsler, 2011). However, the dependency model used by a recommender system relying on association rules can be fully precomputed. Another advantage of association rules is that those with more than one item at the left-hand side (X) take into account higher-order effects between items. In order to define a top-N list for a certain user u_n owning the set of items T_n, the rules $X \rightarrow Y$ for which $X \subseteq T_n$, part of the dependency model, should first be identified. Then, the n unique right-hand-sides (Y) of these rules with the highest confidence are recommended.

Popular and random. The popular recommender method recommends items based on item popularity and the random recommender method simply generates random recommendations. Both methods are included to benchmark against the more complex user- and item based collaborative filtering and recommender systems relying on association rules.

3.4. Model evaluation criteria

In order to evaluate the performance of the analytical techniques, we use the F_1-measure and G-mean. F_1-measure and G-mean are general model evaluation criteria that can be used to evaluate both recommender systems and multi-label classification techniques. According to the framework of Madjarov et al. (2012) multi-label classification performance measures are categorized into bipartition-based and ranking-based measures. Since we are comparing predicted labels with the true labels, we will use bipartition-based measures. Bipartition measures, on their part, are divided into example- based and label- based measures. The former average the results across all users, the latter average across all labels. To compare multi-label classification with recommender systems we have to consider the label dependencies (Debaere et al., 2018). Therefore, we chose example-based measures. In the case of both multi-label classification and recommender systems, most performance measures are based on accuracy, precision and recall (Herlocker et al., 2004). Besides reporting precision and recall separately, we also wanted to incorporate precision and recall into one composite measure. The most widely-used measure is the F_1 measure, which is the harmonic mean between precision and recall (i.e., $\beta = 1$) (Debaere et al., 2018). Another measure, often used in unbalanced settings, is the G-mean (Bogaert, Ballings, & Van den Poel, 2018), which is the geometric mean between precision and recall. F_1 measure and G-mean measure the trade-off between the accuracy and the robustness of a classifier. In the remainder of this section we further elaborate on accuracy, precision, recall, F_1, and G-mean in a multi-label setting (Charte & Charte, 2015).

Just like in binary classification problems, a confusion matrix (Kohavi & Provost, 1998) can be created to represent the classification problem and derive the classification performance, as can be seen in Fig. 1. However, we should consider that in the present

actual \ predicted	positive	negative
positive	TP	FN
negative	FP	TN

Fig. 1. Confusion matrix.

study multiple products can be associated with each customer and we thus deal with a multi-label problem. As such, the confusion matrix is on the user-level, not the total sample level. The true positives (TP) measure how many of the recommended products are actually possessed by a specific user (i.e., correct predictions), the false positives calculate how many of the recommended products are not possessed by a specific user (i.e., incorrect recommendations). The matrix also shows how many of the products that were not recommended should have actually been recommended (i.e., false negatives or FN) and how many are correctly not recommended (i.e., true negatives or TN) (Hahsler, 2011). We note that for recommender systems the g products given (cf. supra) are excluded from the confusion matrix, as the recommender system generates recommendations given these g products (Breese et al., 1998). The confusion matrix is then used to determine to which extent the recommender system succeeded in correctly recommending the remaining products possessed by a customer.

From the confusion matrix, the performance measures accuracy, precision and recall can be derived, on which our performance measures of interest, F_1 measure and G-mean, are based. Accuracy (Eq. (4))[1] is calculated as the Jaccard similarity index between the predicted labels and the true labels (Charte & Charte, 2015; Madjarov et al., 2012). Precision (Eq. (5)) is a measure of exactness describing the proportion of predicted items that are relevant, whereas recall (Eq. (6)) is a measure of completeness which reports the proportion of relevant items that are recommended (He & Garcia, 2009):

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i + FN_i}, \tag{4}$$

$$Precision = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FP_i}, \tag{5}$$

$$Recall = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i}, \tag{6}$$

with N the number of users in the dataset. As can be seen in Eqs. (4)–(6), accuracy, precision and recall are calculated per user (i.e., per confusion matrix) and averaged over all users.

The F measure originated out of van Rijsbergen's E-function (van Rijsbergen, 1979, chap. 7) and can be defined as follows:

$$F\ measure = F_\beta = \frac{\left(\beta^2 + 1\right) * Precision * Recall}{(\beta^2 * Precision) + Recall}, \tag{6}$$

where β determines the weighting between precision and recall. When β equals 1, the F-measure is the harmonic mean of precision and recall and is referred to as F_1 (Baumann, Hochbaum, & Yang, 2019):

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}. \tag{7}$$

[1] Remark that in contrast to the definition of accuracy in a binary setting, the true negatives are not present in the case of multi-label example-based accuracy.

Since we focus on precision and recall, we calculate the G-mean as the geometric mean of precision and recall (Bogaert et al., 2018):

$$G - mean = \sqrt{Precision * Recall}. \tag{8}$$

As the F_1 measure calculates the harmonic mean of precision and recall, the results are less biased towards extreme differences in precision and recall than the G-mean.

3.5. Cross-validation

In order to obtain a reliable estimate of model performance, we use a five-fold cross-validation (5cv) approach (Kohavi, 1995). In 5cv, the complete dataset (D) is randomly divided into five mutually exclusive folds ($D_1, D_2, \ldots\ldots, D_5$) of equal size. The classifier is trained and tested five times, each time ($t \in \{1, 2, \ldots\ldots, 5\}$) using all but one fold ($D \setminus D_t$) as training set and the remaining single fold (D_t) as test set (Kohavi, 1995). This procedure results in five estimates of the F_1 measure and G-mean per classifier. As a measure of overall performance, we describe the median F_1 measure and G-mean over the five different models created by 5cv. In addition, we also include the median absolute deviation (MAD) as a measure of dispersion:

$$MAD = median(\ |y_i - median(y)|\ ). \tag{9}$$

We use the non-parametric Friedman test (Friedman, 1940) with Bonferroni–Dunn post hoc test to determine whether the different multi-label classification techniques and recommender systems are significantly different from each other (Demšar, 2006). Next, these tests are also used to determine if the best performing recommender system and the best performing multi-label classification (i.e., best problem transformation and best algorithm adaptation method) techniques yield significantly different performances. The Friedman test ranks the different classifiers within each fold, assigning 1 to the best performing classifier, 2 to the second best performing classifier, etc. In case of ties the classifiers get the average rank.

If the Friedman statistic is greater than the critical value (at the 0.05 significance level in our case), the null hypothesis, stating that the different algorithms are not significantly different, is rejected. In that case, we perform the Bonferroni–Dunn post hoc test (Dunn, 1961) to compare the different classifiers to a control classifier (i.e., the best performing algorithm). We prefer the more powerful Bonferroni–Dunn test over the Nemenyi post-hoc test (Nemenyi, 1963), since the Bonferroni–Dunn tests adjust the critical value for making $k-1$ comparison, whereas the Nemenyi test controls for making $k(k-1)/2$ comparisons. This adjustment makes the Bonferroni–Dunn test more powerful in this case, since we compare all the classifiers to a control classifier and not to each other (Demšar, 2006). A classifier is significantly different from the control classifier if its average ranks differ by at least the critical difference (CD), defined as follows:

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}, \tag{11}$$

with q_α the critical value for a given significance level α (α is 0.05 in our case), k the number of different algorithms (in our case 5 recommender systems, 14 multi-label classification techniques and 3 best performing methods) and N the number of folds (in our case 5). The critical differences in our study are 2.498, 7.648 and 1.418 for the recommender systems, multi-label classification techniques and the comparison between the best performing methods respectively.

To determine whether the problem transformation methods binary relevance and classifier chains differ significantly from each other, we use the non-parametric Wilcoxon signed-ranks

Table 5
Summary results of multi-label algorithms and recommender systems.

Multi-label algorithms			
	F_1 measure	*G*-mean	Diversity
General	CC_AB > BR_NN > BR_RF > BR_AB > CC_NN	BR_RF > BR_NN > BR_AB > CC_AB > CC_NN	–
Problem transformation			
Binary relevance	BR_NN > BR_RF > BR_AB	BR_RF > BR_NN > BR_AB	–
Classifier chains	CC_AB > CC_NN > CC_DT	CC_AB > CC_NN > CC_DT	
Algorithm adaptation	RF > rFerns	RF > rFerns	
Significant differences	Equal to CC_AB: BR_NN, BR_RF, CC_NN, CC_DT, BR_DT, RF	Equal to BR_RF: CC_AB, CC_NN, CC_DT, BR_DT, RF, BR_NB	–
Recommender systems			
General	UBCF > IBCF > POPULAR > AR	UBCF > IBCF > POPULAR > AR	IBCF > UBCF > AR > POPULAR
Significant differences	Equal to UBCF: IBCF, POPULAR	Equal to UBCF: IBCF, POPULAR	–

Note: BR_DT = binary relevance decision trees, BR_NB = binary relevance naive Bayes, BR_rFerns = binary relevance random ferns, BR_AB = binary relevance adaboost, BR_RF = binary relevance random forest, CC_DT = classifier chains decision trees, CC_NB = classifier chains naive Bayes, CC_AB = classifier chains adaboost, rFerns = random ferns, RF = random forest, UBCF = user-based collaborative filtering, IBCF = item-based collaborative filtering, AR = association rules, AR = association rules, POPULAR = popular recommender method.

Table 6
5cv median precision, recall, accuracy, F_1 measure and *G*-mean for multi-label classification techniques.

Algorithm	Optimal parameter combination	Precision	Recall	Accuracy	F_1 measure	*G*-mean
(a) PT methods						
BR_DT (F_1)	$cp = 0.001$	0.7161	0.3991	0.3549	0.5125	–
BR_DT (*G*-mean)	$cp = 0.01$	0.7062	0.3498	0.3151	–	0.4970
BR_NB		0.4182	0.5855	0.2983	0.4890	0.4962
BR_NN	***size* = 5, *decay* = 0.1**	**0.7197**	**0.4154**	**0.3705**	**0.5269**	**0.5469**
BR_rFerns		*0.1938*	*0.7626*	*0.1836*	*0.3090*	*0.3847*
BR_AB		0.7212	0.4096	0.3655	0.5226	0.5438
BR_RF		**0.7301**	**0.4110**	**0.3688**	**0.5257**	**0.5478**
CC_DT (F_1)	cp = 0.001	0.6528	0.4318	0.3722	0.5193	–
CC_DT (*G*-mean)	$cp = 0.01$	0.6608	0.3730	0.3298	–	0.4963
CC_NB		0.1399	0.6374	0.1312	0.2294	0.2986
CC_NN	$size = 5,\ decay = 0.1$	0.6934	0.4254	0.3666	0.5221	0.5367
CC_rFerns		0.2090	0.7246	0.1943	0.3246	0.3897
CC_AB		***0.6614***	***0.4469***	***0.3860***	***0.5330***	***0.5432***
CC_RF		0.7204	0.3361	0.3049	0.4576	0.4908
(b) AA methods						
rFerns		0.1945	0.7621	0.1841	0.3099	0.3852
RF		***0.7428***	***0.3836***	***0.3494***	***0.5059***	***0.5338***
Random		0.0688	0.0716	0.0719	0.0713	0.0715

Note: PT methods = Problem Transformation methods, AA methods = Algorithm Adaptation methods, BR_DT = binary relevance decision trees, BR_NB = binary relevance naive Bayes, BR_rFerns = binary relevance random ferns, BR_AB = binary relevance adaboost, BR_RF = binary relevance random forest, CC_DT = classifier chains decision trees, CC_NB = classifier chains naive Bayes, CC_rFerns = classifier chains random ferns, CC_AB = classifier chains adaboost, CC_RF = classifier chains random forest, rFerns = random ferns, RF = random forest. The figures in bold reflect the best algorithm for F_1 and G-mean for problem transformation (i.e., binary relevance and problem transformation) and algorithm adaptation methods.

test (Wilcoxon, 1945). The Wilcoxon signed-ranks test's parametric counterpart is the paired *t*-test, which is considered more powerful than the Wilcoxon signed-rank test when its assumptions are met. However, we cannot guarantee that the differences in performance between the two problem transformation methods are normally distributed, since we only have a sample size of 6 (Demšar, 2006). In that case, the Wilcoxon signed-rank test is preferred according to Demšar (2006).

4. Results

In this section we first discuss the performance results of multi-label algorithms and recommender systems independently. Next, we compare both methods and perform additional robustness checks. Table 5 summarizes the main findings for multi-label classifiers and recommender systems.

4.1. Multi-label classification performance

Table 6 shows the median F_1 measure and *G*-mean values per multi-label classification algorithm, with their corresponding median precision and recall values and optimal parameter combination where applicable. Each algorithm contains either the predefined parameter values or the optimal parameter combination obtained by cross-validation (column 'Optimal parameter combination' in Table 6). We refer to Appendix C for detailed results of the parameter tuning. We note that for algorithms where the optimal tuning parameters differed in terms of F_1 measure and *G*-mean, we include the results for both optimal tuning parameters (e.g., BR_DT (F_1) and BR_DT (*G*-mean)). From Table 6, it appears that multi-label classification techniques yield good results in predicting cross-sell opportunities when compared to a random classifier. The F_1 measure ranges from 22.94% to 53.30%[2] and the *G*-mean from 29.86% to 54.78%. The corresponding values of precision, recall and accuracy range from 13.99% to 74.28%, from 33.61% to 76.26%, and from 13.12% to 38 respectively. These results are also in line to the values obtained in previous benchmark studies with a comparable number of labels (Montañes et al., 2014).

Table 6 also shows that classifier chains in combination with adaboost (CC_AB) appears to be the best performing multi-label classification technique and also the best problem transformation

[2] The results of the random classifier are excluded in this range.

method for the F_1 measure (in bold), followed by binary relevance combined with neural networks (BR_NN) and random forest (BR_RF) and binary relevance in combination with adaboost (BR_AB). Further, classifier chains in combination with neural networks (CC_NN) and classifier chains and binary relevance in combination with decision trees (CC_DT and BR_DT) also obtain good performance. We remark that multi-label random forest (RF) is the best algorithm adaptation method (in bold). Conversely, all techniques based on the random ferns algorithm (i.e., classifier chains and binary relevance in combination with random ferns (CC_rFerns, BR_rFerns) and multi-label random ferns (rFerns)) have poor performance. Classifier chains combined with naive Bayes (CC_NB) performs worst. As for the *G*-mean, BR_RF performs better than BR_NN, CC_AB, and BR_AB. The best algorithms for each problem transformation and algorithm adaption method are in bold in Table 4. BR_NN for binary relevance in terms of F_1 and BR_RF in terms of *G*-mean, CC_AB for classifier chains for both F_1 and *G*-mean and RF for algorithm adaptation methods for both F_1 and *G*-mean. The best algorithms in terms of precision, recall, and accuracy are in italics in Table 6. RF for precision, BR_rFerns for recall, and CC_AB for accuracy. We note that the values of the *G*-mean are higher than the F_1 measure values, since the F_1 measure is less biased due to extreme precision and/or recall values.

When analyzing our results in detail, we remark the following elements. First, for the precision and recall values we observe that the worst performing algorithms (i.e., BR_rFerns, CC_NB, CC_rFerns and rFerns) are characterized by a high recall and a very low precision. This means that these algorithms succeed in recommending most of the relevant items (high *recall*), even though they recommend a large number of items which are mostly irrelevant (low *precision*). In other words, these algorithms have for a given number of true positives a large number of false positives (low *precision*) or a small number of false negatives (high *recall*). The top performing algorithms (i.e., CC_AB, BR_RF, CC_NN, BR_NN) combine a high precision with an acceptable recall. This means that for a given number of true positives these algorithms have a small number of false positives and that most of the predicted items were relevant (high *precision*). However, for a given number of true positives they have a moderate number of false negatives and thus there are still some relevant products that are not suggested (moderate *recall*). One might argue that the worst performing algorithms (i.e., BR_rFerns, CC_NB, CC_rFerns and rFerns) provide more complete recommendations. For a given number of true positives, these algorithms have a smaller number of false negatives and thus they are able to recommend more relevant products. However, a high precision with a high number of false positives is not desired. If there are a lot of false positives, users will then consider the recommendations as spam, which in turn will decrease the overall satisfaction of the users (Bogaert, Ballings, Hosten, & Van den Poel, 2017). So, excelling in precision or recall is not sufficient, good classifiers perform well on both (Probst et al., 2017), which is captured by the F_1 measure and the *G*-mean. Hence, an algorithm that scores high on the F_1 measure (and the *G*-mean) detects a high number of relevant items and minimizes the number of false alarms. The F_1 measure indicates that an algorithm is both accurate and robust. In an ideal situation a high precision is combined with a high recall. If this is not the case, the practitioner must choose between a high value for precision and a moderate recall or vice versa. In the case of our top performing multi-label classifiers (i.e., CC_AB, BR_RF, CC_NN, BR_NN), the algorithms perform better on precision, meaning that they are more focused on detecting truly relevant items, instead of recommending all relevant items. If the practitioner would prefer to have a higher recall but still retain a moderate precision, the best algorithms are BR_NB and CC_DT (in italics in Table 6). Disregarding the fact that the algorithm is not amongst the top performers across all performance measures, it still provides the best balance between precision and recall if the practitioners favors recall more than precision.

Second, our benchmark experiment confirms the dominance of adaboost as base classifier over the random forest base classifier (Probst et al., 2017). However, the finding that CC_AB and CC_RF consistently outperform the algorithm adaptation methods RF and rFerns is not confirmed: CC_AB indeed outperforms RF and rFerns, but CC_RF performs worse than RF. However, we do find that BR_RF outperforms RF.

Third, our findings indicate that besides adaboost and random forest, neural networks also perform well as a base classifier for binary relevance and classifier chains.

Fourth, the inferior performance of CC_RF can be explained by the fact that the combination of chaining and an ensemble of trees is too complex and leads to overfitting. This assumption is confirmed by the fact that one single decision tree coupled with classifier chains (i.e., CC_DT) outperforms an ensemble of trees coupled with classifier chains (i.e., CC_RF) and BR_RF does as well.

Fifth, we find evidence that a decision tree is an appropriate base classifier for binarized multi-label problems consisting of large datasets with a smaller number of features (Madjarov et al., 2012). We note a good performance both in combination with binary relevance and classifier chains.

Sixth, the bad performance of all techniques with the random ferns classifier (i.e., BR_rFerns, CC_rFerns and rFerns) can be explained by the classifier's implementation. The classifier not only randomly chooses a variable on which a split is based, but also randomly defines this variable's splitting point. As such, the random ferns classifier cannot handle too many features resulting in a low predictive value (Probst et al., 2017). Notice that in terms of recall random ferns is always amongst the best algorithms (in italics in Table 6). However, when looking at F_1 measures, *G*-mean, and accuracy we see that the performance drops dramatically, indicating that rFerns techniques are not well-balanced (between precision and recall for F_1 measure and *G*-mean, and between recall and specificity for accuracy).

Seventh, we elaborate on the bad performance of CC_NB. The classifier chains method extends each label's predictor variables with the previous labels in order to take into account label dependencies, which conflicts with the naive Bayes classifier's independence assumption.

Finally, we stress the competitive overall performance of random forest as an algorithm adaption method. Whereas random forest is not the top performer in terms of the F_1 measure and *G*-mean, we see that the performance is reasonable across all performance measures and in terms of precision it is even the top performer. Hence, random forest comes up with a lot of relevant recommendations for the user.

To investigate whether the multi-label classification techniques are significantly different from each other, we apply the Friedman test. The Friedman test indicates significant differences across the multi-label classification techniques for the F_1 measure ($\chi_F^2 = 63.5371,\ p < 0.001$) and *G*-mean ($\chi_F^2 = 63.9943,\ p < 0.001$), so the Bonferroni–Dunn post-hoc test is used to determine which algorithms differ significantly from the best performing multi-label classification technique (i.e., CC_AB for F_1 measure and BR_AB for *G*-mean). Fig. 2 contains a graphical representation of the results of the Bonferroni–Dunn post-hoc (De Weerdt, De Backer, Vanthienen, & Baesens, 2011). The horizontal axis in Fig. 2 represents the average rank of a multi-label classification technique across the five folds. Each multi-label classification technique is depicted by a horizontal line, of which the left end represents the average ranking and the length is equal to the critical difference obtained by the Bonferroni–Dunn post-hoc test at the 95% confidence level (i.e., 7.648 for both F_1 measure and *G*-mean). Multi-label classification techniques are significantly outperformed by CC_AB (F_1 measure)

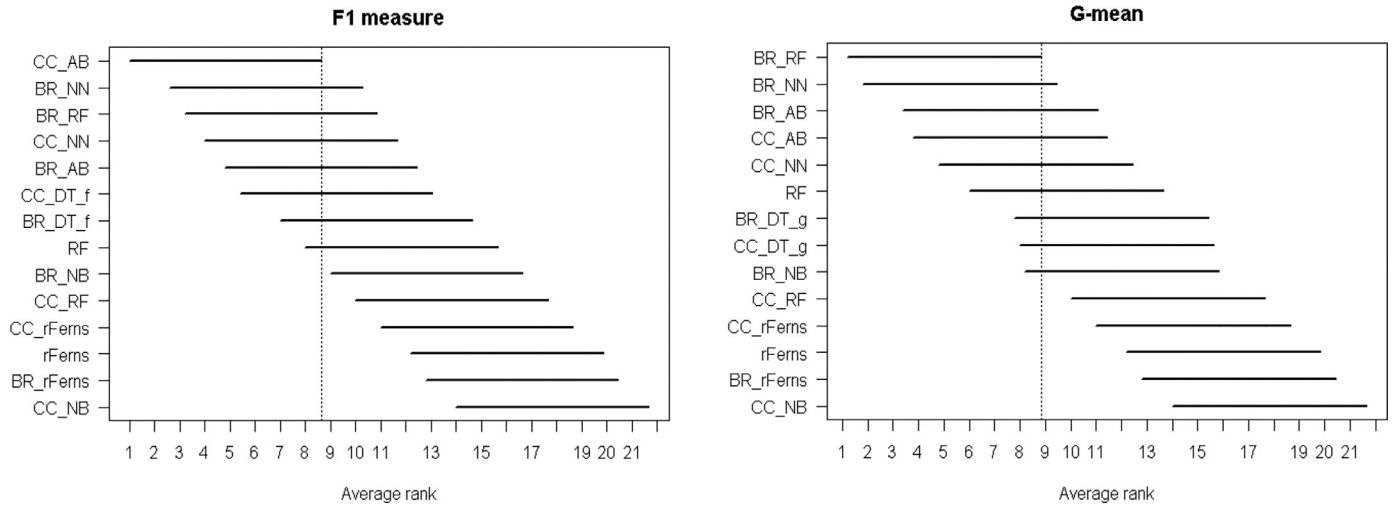

Fig. 2. Plot of Bonferroni–Dunn post-hoc test for F_1 measure (left) and G-mean (right) of multi-label classification techniques, with critical difference of 7.648. The dashed vertical line represents the 95% confidence level.

Table 7
MAD of precision, recall, accuracy, F_1 measure and G-mean obtained through 5cv for multi-label classification techniques.

Algorithm	Optimal parameter combination	MAD of precision	MAD of recall	MAD of accuracy	MAD of F_1 measure	MAD of G-mean
(a) PT methods						
BR_DT (F_1)	$cp = 0.001$	0.0018	0.0016	0.0012	0.0008	–
BR_DT (G-mean)	$cp = 0.01$	0.0024	0.0021	0.0024	–	0.0013
BR_NB		0.0009	0.0014	0.0007	0.0005	0.0001
BR_NN	$size = 5,\ decay = 0.1$	0.0006	0.0005	0.0007	0.0007	0.0010
BR_rFerns		0.0003	0.0006	0.0001	0.0002	0.0003
BR_AB		0.0009	0.0004	0.0002	0.0004	0.0002
BR_RF		0.0012	0.0002	0.0002	0.0003	0.0004
CC_DT (F_1)	$cp = 0.001$	0.0024	0.0012	0.0005	0.0004	–
CC_DT (G-mean)	$cp = 0.01$	0.0027	0.0012	0.0007	–	0.0011
CC_NB		0.0017	0.0008	0.0016	0.0022	0.0016
CC_NN	$size = 5,\ decay = 0.1$	0.0128	0.0101	0.0043	0.0052	0.0023
CC_rFerns		0.0029	0.0032	0.0018	0.0027	0.0006
CC_AB		0.0003	0.001	0.0010	0.0009	0.0008
CC_RF		0.0016	0.0002	0.0004	0.0012	0.0013
(b) AA methods						
rFerns		0.0005	0.0012	0.0004	0.0006	0.0002
RF		0.0013	0.0007	0.0006	0.0004	0.0002

Note: PT methods = Problem Transformation methods, AA methods = Algorithm Adaptation methods, BR_DT = binary relevance decision trees, BR_NB = binary relevance naive Bayes, BR_rFerns = binary relevance random ferns, BR_AB = binary relevance adaboost, BR_RF = binary relevance random forest, CC_DT = classifier chains decision trees, CC_NB = classifier chains naive Bayes, CC_rFerns = classifier chains random ferns, CC_AB = classifier chains adaboost, CC_RF = classifier chains random forest, rFerns = random ferns, RF = random forest.

and BR_RF (G-mean) when their difference in average ranking exceeds the critical difference (i.e., positioned at the right side of the dashed vertical line in Fig. 2). We can then identify:two groups: algorithms that do not differ significantly from the best performing algorithm CC_AB (i.e., BR_NN, BR_RF, CC_NN, CC_DT, BR_DT, and RF) and significantly worse performing algorithms (i.e., BR_NB, CC_RF, CC_rFerns, rFerns, BR_rFerns and CC_NB). For the G-mean, BR_RF is the best performing algorithm. In that case, BR_NB is not significantly different from BR_RF.

Table 7 displays the median absolute deviation (MAD). The MAD ranges from 0.0002 to 0.0052 for F_1 measure, from 0.0002 to 0.0023 for G-mean. All MAD values are low, meaning that all fourteen multi-label classification techniques produce stable results. Overall, BR_rFerns produces the most stable results. Regarding the top performing algorithms, BR_RF has the lowest MAD values.

More generally, it can be questioned if there is a significant difference between the binary transformation methods binary relevance and classifier chains. According to the Wilcoxon signed-rank test, there is no significant difference between both methods for F_1 measure ($V = 12,\ p = 0.8438$) and G-mean ($V = 18,\ p = 0.1563$). Our experiment does not find evidence that the classifier chains method dominates the binary relevance method (Read et al., 2009).

In conclusion, CC_AB and BR_RF are the best performing multi-label classification techniques for respectively the F_1 measure and G-mean. As such, the ability of the adaboost and random forest algorithm to reduce variance (Bauer & Kohavi, 1999) applies to a multi-label problem context as well. Besides these techniques, CC_NN, BR_NN, CC_DT, BR_DT and BR_AB also perform well. The dominance of tree-based structures (i.e., the adaboost, decision tree and random forest base classifiers are tree-based structures) in our dataset is thus clearly shown. When looking at computational complexity, BR_DT and CC_DT are the best choice for the non-ensemble methods and BR_RF is preferred as ensemble method.

4.2. Recommender system performance

Table 8 shows the optimal median F_1 measure and G-mean values per recommender algorithm when using the optimal tuning parameters. If the tuning parameters were different when optimizing F_1 measure or the G-mean, we show the tuning results

Table 8
5cv median precision, recall, accuracy, F_1 measure and G-mean for recommender systems.

Algorithm	Optimal parameter combination	Precision	Recall	Accuracy	F_1 measure	G-mean
UBCF (F_1)	$g = 2,\ nn = 100,\ n = 4$	**0.4122**	**0.4322**	**0.2620**	**0.4220**	–
UBCF (G-mean)	$g = 2,\ nn = 100,\ n = 5$	***0.3729***	***0.4874***	***0.2634***	–	***0.4264***
IBCF (F_1)	$g = 2,\ k = 50,\ n = 5$	0.3633	0.4713	0.2547	0.4103	–
IBCF (G-mean)	$g = 2,\ k = 50,\ n = 6$	*0.3343*	*0.5205*	*0.2530*	–	*0.4172*
AR (F_1)	$g = 3,\ s = 0.01,\ c = 0.6,\ n = 4$	0.3875	0.2353	0.1636	0.2929	–
AR (G-mean)	$g = 2,\ s = 0.01,\ c = 0.6,\ n = 4$	*0.5337*	*0.1621*	*0.1367*	–	*0.2941*
POPULAR	$g = 2,\ n = 4$	0.3838	0.3938	0.2330	0.3887	0.3888
RANDOM	$g = 2,\ n = 8$	0.0487	0.0989	0.0337	0.0653	0.0694

Note: UBCF = user-based collaborative filtering, IBCF = item-based collaborative filtering, AR = association rules, AR = association rules, POPULAR = popular recommender method, RANDOM = random recommendation.

for both optimal parameter values (e.g., UBCF (F_1) and UBCF (G-mean)). The values in bold in Table 8 highlight the best algorithms for F_1 measure and G-mean, whereas the values in italics indicate the best algorithms based on precision, recall, and accuracy. Each algorithm contains the optimal parameter combination obtained by parameter tuning using cross-validation (column 'Optimal parameter combination' in Table 8). We refer the reader to Appendix C for the detailed performance of the parameter tuning. The optimal parameter combination can differ between F_1 measure and G-mean for the same algorithm, which is the case for user-based collaborative filtering (UBCF), item-based collaborative filtering (IBCF), and association rules (AR). Based on Table 8, we can conclude that recommender systems obtain reasonable performance in predicting the financial products a customer will buy next. The F_1 measure ranges from 29.29% to 42.20% and the G-mean from 29.41% to 42.64%, while excluding the random recommender method (RANDOM). The corresponding values of precision, recall, and accuracy range from 33.14% to 53.37%, from 23.53% to 52.05%, and from 13.67% to 26.34% while excluding RANDOM. The results are in line with the values obtained by Hahsler (2011) on the MSWeb dataset containing web click-stream data. Concerning these precision and recall values, we observe that the popular recommender method (POPULAR) and UBCF (optimizing the F_1 measure) algorithms have well balanced precision and recall values. This means that these algorithms are able to identify a moderate number of relevant items while keeping the number of false positives reasonably low. Recommender systems using association rules (AR) clearly show a higher precision than recall, in contrast to the other algorithms, at which recall exceeds precision. AR, optimizing the G-mean, score better on precision than recall which can be explained by the fact that AR not always recommend the n number of products. This happens when the g given items of a specific user are not part of enough relevant association rules. AR are better in correctly recommending relevant items than in making sure that all relevant items are recommended. For IBCF and UBCF optimizing the G-mean, higher n's result in a higher recall, because more recommended items augment the probability of recommending all relevant items. However, higher n's result as well in a lower precision, meaning that non-relevant items are predicted too. One could argue that the UBCF optimizing the G-mean provides more complete recommendations since more relevant items are detected. Hence, if the practitioner desires a balance between precision and recall, UBCF optimizing the F_1 measure should be chosen. If a practitioner wants to detect the most relevant items but keep the number of false alarms as low as possible, he/she should optimize UBCF according to the G-mean. If a practitioner wants to detect the maximum number of relevant items, IBCF optimizing the G-mean is best option.

From Table 8, it appears that overall UBCF is the best performing recommender approach, followed by IBCF, POPULAR, AR and RANDOM. Indeed, UBCF yields a slightly superior performance in comparison with IBCF (Breese et al., 1998). In our case, AR are not

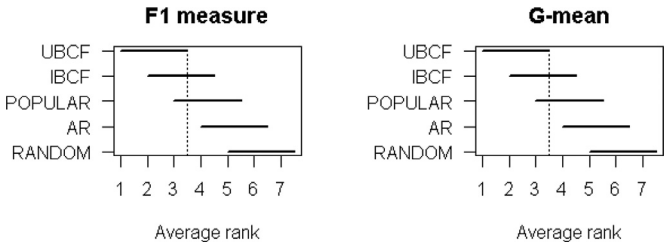

Fig. 3. Plot of Bonferroni–Dunn post-hoc test for F_1 measure (left) and G-mean (right) of recommender systems, with critical difference of 2.498. The dashed vertical line represents the 95% confidence level.

able to outperform UBCF, in contrast to the beliefs of Mobasher et al. (2001). When disregarding the lower performance of AR, the obtained ranking corresponds to previous results (Zhao & Cen, 2013, chap. 5).

To check whether the recommender approaches are significantly different from each other, we apply the Friedman test. The Friedman test indicates significant differences across the recommender approaches for the F_1 measure ($\chi_F^2 = 20,\ p < 0.001$) and G-mean ($\chi_F^2 = 20,\ p < 0.001$), so the Bonferroni–Dunn post-hoc test is used to determine which algorithms significantly differ from the best performing recommender approach (i.e., UBCF). A graphical representation of the results of the Bonferroni–Dunn post-hoc test is presented in Fig. 3. In the case of recommender systems, the critical difference at the 95% confidence level obtained by the Bonferroni–Dunn post-hoc test is 2.498 for both F_1 measure and G-mean. In Fig. 3, we can see that IBCF and POPULAR are not significantly different from UBCF and that AR and RANDOM yield a significantly worse performance than UBCF, for both the F_1 measure and G-mean. We remark that POPULAR, which only recommends the six most popular products, is not significantly worse than the more complex UBCF in terms of F_1 measure and G-mean. We also notice that the average ranks results for F_1 and G-mean measures are identical (see Fig. 3(a) and (b)), since all recommender approaches are equally ranked across all five folds for both F_1 measure and G-mean.

Table 9 reports the median absolute deviation (MAD) of the F_1 measure and G-mean across all five folds as measure of dispersion. The MAD ranges from 0.0001 to 0.0010 for F_1 measure and from 0.0001 to 0.0014 for G-mean. The MAD values are low, meaning that all recommender approaches produce stable results. Overall, the MAD results confirm the superiority of UBCF over IBCF in terms of stability of precision, recall, accuracy, F_1 measure and G-mean.

Besides accuracy, the level of diversity offered to the users is also an important characteristic of a recommender system (Geuens et al., 2018). Diversity measures the number of distinct recommended products over all recommendations per algorithm (Adomavicius & Kwon, 2012). A low diversity would imply that the

Table 9
MAD of precision, recall, accuracy, F_1 measure and G-mean obtained through 5cv for recommender systems.

Algorithm	Optimal parameter combination	MAD of precision	MAD of recall	MAD of accuracy	MAD of F_1 measure	MAD of G-mean
UBCF (F_1)	$g = 2,\ nn = 100,\ n = 4$	0.0009	0.0003	0.0006	0.0006	
UBCF (G-mean)	$g = 2,\ nn = 100,\ n = 5$	0.0001	0.0009	0.0001		0.0005
IBCF (F_1)	$g = 2,\ k = 50,\ n = 5$	0.0009	0.0010	0.0015	0.0010	
IBCF (G-mean)	$g = 2,\ k = 50,\ n = 6$	0.0010	0.0008	0.0001		0.0008
AR (F_1)	$g = 3,\ s = 0.01,\ c = 0.6,\ n = 4$	0.0026	0.0001	0.0007	0.0009	
AR (G-mean)	$g = 2,\ s = 0.01,\ c = 0.6,\ n = 4$	0.0055	0.0008	0.0001		0.0014
POPULAR	$g = 2,\ n = 4$	0.0005	0.0001	0.0001	0.0003	0.0003
RANDOM	$g = 2,\ n = 8$	0.0001	0.0001	0.0001	0.0001	0.0001

Note: UBCF = user-based collaborative filtering, IBCF = item-based collaborative filtering, AR = association rules, AR = association rules, POPULAR = popular recommender method, RANDOM = random recommendation.

Table 10
Comparison of most frequently recommended products between UBCF optimizing F_1 measure (a), IBCF optimizing F_1 measure (b) and POPULAR (c).

(a)		(b)		(c)	
Product ID	Frequency	Product ID	Frequency	Product ID	Frequency
product_1	13,516	product_1	13,172	product_3	16,193
product_4	10,291	product_3	12,719	product_4	16,193
product_9	9508	product_4	12,486	product_9	16,120
product_3	9400	product_8	12,329	product_1	15,499
product_8	9130	product_9	12,229	product_8	9907
product_2	3575	product_2	5671	product_2	3368

same items are recommended. As the recommendation list will determine the user's overall impression of the whole recommender system, a monotonous list might disappoint the users. This in turn lowers the overall evaluation of the recommender system and decreases customer satisfaction (Pu, Chen, & Hu, 2011). Especially in the case of cross-selling it is important that users are presented with a diverse list, since this increases the probability that the user will buy a new product (Geuens et al., 2018).

In terms of diversity RANDOM is the absolute winner with all 83 products being recommended, followed by IBCF with 65 distinct products (78% of all products) and UBCF with 44 distinct products (53% of all products). AR recommends 7 distinct products (9% of all products) and POPULAR ends last with 6 products or 8% of all products. Remark that the higher diversity of IBCF and UBCF does not result in a significantly better performance than POPULAR, which has the lowest recommendation diversity.

Next, we compare the most frequently recommended products between the best performing recommender approaches, namely UBCF, IBCF and POPULAR. UBCF's and IBCF's most frequently recommended products correspond to the 6 most popular products, as can be seen in Table 10. The most popular products are also the most frequently bought products, so it is obvious that these products are also recommended most by both collaborative filtering approaches, certainly in a dataset containing binary purchase information instead of ratings. Popular items have the most information (i.e., most ones), so they can be recommended to more users (Adomavicius & Kwon, 2012). Due to a non-disclosure agreement, we do not have the product names in Table 10.

In summary, UBCF is the best performing recommender approach, with a sufficient recommendation diversity. However, IBCF and POPULAR do not perform significantly worse. IBCF offers high recommendation diversity and is computationally less intensive than UBCF. POPULAR is a straightforward recommendation approach: it yields competitive results to UBCF by only recommending the six most popular products.

4.3. Performance comparison

It can be concluded that CC_AB and BR_RF are the best problem transformation multi-label classification techniques for F_1 measure and G-mean respectively and that RF is the best algorithm adaptation multi-label classification technique for both F_1 measure and G-mean. Further, UBCF is the best recommender approach for both F_1 measure and G-mean. The three methods can be ranked for F_1 measure and G-mean as follows: CC_AB, RF, UBCF and BR_RF, RF, UBCF respectively.

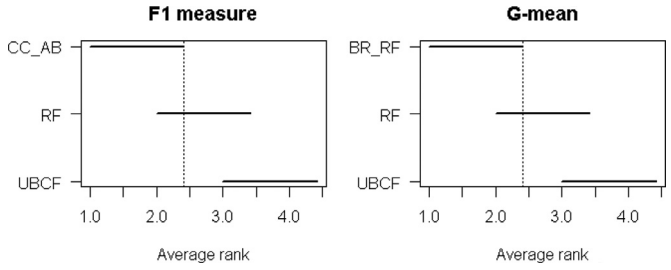

Fig. 4. Plot of Bonferroni–Dunn post-hoc test for F_1 measure (left) and G-mean (right) of best performing techniques, with critical difference of 1.418. The dashed vertical line represents the 95% confidence level.

To examine whether the three best performing multi-label classification and recommender approaches are significantly different from each other, we apply the Friedman test. The Friedman test indicates significant differences across the best performing approaches for the F_1 measure ($\chi_F^2 = 10,\ p = 0.0067$) and G-mean ($\chi_F^2 = 10,\ p = 0.0067$), so the Bonferroni–Dunn post-hoc test is used to determine which algorithms significantly differ from the best performing technique (i.e., CC_AB for F_1 measure and BR_RF for G-mean). A graphical representation of the results of the Bonferroni–Dunn post-hoc test is presented in Fig. 4. In this case, the critical difference at the 95% confidence level obtained by the Bonferroni–Dunn test is 1.418 for F_1 measure and G-mean. In Fig. 4, we can see that RF within this comparison performs not significantly worse than the best performing technique and that UBCF shows a significantly worse performance, for both F_1 measure and G-mean.

In summary, we can state that multi-label classification techniques are in this case the best option to determine cross-sell opportunities. It must be mentioned that multi-label classification techniques can benefit from predictor variables to base their predictions on. However, within our dataset, the 38 predictor variables

only encompass basic social-demographic and behavioral variables. Other studies in the financial services such as Prinzie and Van den Poel (2006) and Van den Poel and Larivière (2004) have more detailed information about product-specific ownership (e.g., names of the products and the product categories) and socio-demographics of their customers (e.g., education and social status). For example, Van den Poel and Larivière (2004) found that more educated and wealthy people have higher customer loyalty which in turn makes them better targets for cross-selling opportunities. Hence, it might be that the performance of our multi-label algorithms can still be improved when more detailed product-specific and customer-specific information would be included.

4.4. Robustness checks

To check whether the results are consistent, we performed the following analyses: we ran the analysis on a subsample of 15,000 customers and we reran the analysis on the full data set with a sparsity level of 0.10%. We summarize the impact of the smaller subsample on the results. We refer the reader to the supplementary materials (Appendix D) for the detailed results. For multi-label classifiers the results slightly differ. First, the results still indicate that classifier chains with adaboost (CC_AB) is the best technique in terms of F_1 measure. However, in terms of G-mean binary relevance with adaboost is the top performer (in contrast to binary relevance with random forest). The worst performing algorithms remain all variations of random ferns and classifier chains with naive Bayes. For recommender systems the top performing technique is still user-based collaborative filtering for F_1 and G-mean. Overall, the absolute performance values on the subsample are smaller than the results on the sample data. This means that the subsample overestimates the performance of the recommender systems.

We also look at the effect of lowering the sparsity level to 0.10% (see Appendix E for more detailed results). A first thing that we notice is that the number of observations increases to 2200,595 and the number of items to 132 (as opposed to 2074,748 observations and 83 items). For multi-label algorithms the results slightly differ. First, the results show that CC_AB is still the best technique in terms of F_1 measure. However, the second best measure was BR_DT, followed by BR_NN and BR_RF (compared to BR_NN, BR_RF and BR_DT). The worst performing algorithm was still CC_NB, however, the performance significantly dropped from 0.2294 to 0.1234. For G-mean the best algorithm (BR_RF) as well as the ranking of the algorithms remained the same. For recommender systems UBCF remained the top performer, followed by IBCF and the POPULAR method. In general we can say that the performance of both multi-label algorithms and recommender systems is around 2 percentage point lower when decreasing the sparsity level from 0.25% to 0.10%. Hence, it is harder to classify instances correctly when the data is sparser, but sparsity level does not significantly influence the ranking of the best algorithms.

5. Conclusion

In this paper, we evaluated recommender systems and multi-label classification techniques in the financial services sector. Our contributions to literature are threefold. First, we tested several multi-label classification techniques, both problem transformation and algorithm adaptation methods. As problem transformation methods, we included the binary relevance and classifier chains methods in combination with several base classifiers. The investigated algorithm adaptation methods are multi-label random forest and multi-label random ferns. These techniques obtained satisfactory results in determining cross-sell opportunities. Classifier chains in combination with adaboost was the top performing algorithms for F_1 measure and binary relevance in combination with random forest for G-mean, with a median cross-validated F_1 measure of 53.30% and G-mean of 54.32%. Further, multi-label random forest, classifier chains and binary relevance combined with neural networks and decision trees obtained good performance. Classifier chains combined with naive Bayes and random ferns performed poorly. We remark that the seven first mentioned algorithms performed equally well in statistical terms, extended with binary relevance combined with naive Bayes for the G-mean.

Second, we investigated the usability of several well-known recommender systems, being user- and item-based collaborative filtering, recommender systems using association rules and the popular and random recommender method. These recommender systems were able to yield reasonable performance in identifying the products a customer will buy next. User-based collaborative filtering was the top performing algorithm in terms of F_1 measure and G-mean, with a median cross-validated F_1 measure of 42.20% and G-mean of 42.64%, followed by item-based collaborative filtering, the popular recommender method, recommender systems relying on association rules and the random recommender method. We note that the top three recommender systems performed equally well in statistical terms.

Third, we compared the best multi-label classification techniques and the best recommender system in order to determine the optimum approach for identifying cross-sell opportunities. For the multi-label classification techniques, we included the best problem transformation and the best algorithm adaptation method. Based on this comparison, classifier chains with adaboost and binary relevance with random forest for F_1 measure and G-mean respectively are the overall top performing algorithms, followed by multi-label random forest and user-based collaborative filtering. Both multi-label techniques performed equally well in statistical terms.

In sum, classifier chains with adaboost and binary relevance with random forest are the best multi-label classification techniques for F_1 measure and G-mean respectively, and also the best problem transformation methods. Multi-label random forest is the best multi-label algorithm adaptation method. Next, user-based collaborative filtering is the best recommender system.

6. Practical implications

Our results provide important recommendations for financial services providers, who are interested in the most effective cross-sell methods to maximally retain their customers and enhance their sales. To highlight the practical relevance of our results, we introduce two scenarios decision makers are faced with when implementing a cross-sell strategy.

In the first scenario, there is only purchase history information available. In this case, we recommend the financial services providers to use recommender systems for cross-sell purposes, and more specifically user-based collaborative filtering. However, our results prove that item-based collaborative filtering and the popular recommender method do not perform significantly worse than user-based collaborative filtering. User-based collaborative filtering remains the preferred option for financial services providers looking for the overall best performing approach. However, item-based collaborative filtering has a lower computational complexity and a higher recall than user-based collaborative filtering. Moreover, item-based collaborative filtering can be praised for its recommendation diversity, which makes it an ideal method for financial services providers that attach great importance to highly personalized and diversified recommendations. Hence, decision makers who are interested in recommending as much distinct products as possible should choose item-based collaborative filtering. The popular recommender method, on its part, requires the least computational effort of the three methods and only

recommends the most popular products, which correspond to the most frequently recommended products by user- and item-based collaborative filtering. Hence, the popular recommender method is characterized by a low recommendation diversity. Given the popular recommender method's traits, it is the ideal method for financial services providers having a limited marketing budget to develop cross-sell actions.

In the second scenario, auxiliary data (e.g., customer characteristics) are available, and can be translated into a range of predictor variables. In this scenario, multi-label classification techniques are preferred, and more specifically classifier chains in combination with adaboost or binary relevance in combination with random forest. Several other multi-label classification techniques however, do not perform significantly worse than these methods, of which classifier chains and binary relevance combined with decision trees are interesting when computation time is important. We also stress that the application area of multi-label classification techniques is thus not limited to cross-sell: these techniques are also suited for customer acquisition. Imagine someone who wants to buy a house and consults several financial services providers for a loan. Since this person consults multiple financial services providers, the chance that the provider will actually sell a loan is relatively small. If the provider can recommend other financial products that might interest the prospect (based on the available information of this prospect), he might have a good chance to gain a new customer. This scenario will become more and more important, since most people do extensive comparisons before buying a financial product.

7. Limitations and future research

A first limitation is related to the dataset, which does not contain all the desired and relevant information. First, a wider range of predictor variables could have resulted in superior multi-label classification models. Examples of relevant additional predictor variables are more profound socio-demographic information (e.g., income, household size, and education level), more thorough consumption information (e.g., share of wallet, sales channel,) and customer satisfaction information. Second, the dataset lacked detailed product information, which would have given us the possibility to set up a content-based recommender system. Content-based recommender systems require product characteristics to be able to construct a profile for each product. Even though the performance of a content-based recommender system is unpredictable in a financial services context, it is something worth investigating. A content-based recommender system could as well be used, together with collaborative filtering, into a hybrid approach.

The second direction for future research is to conduct a real-life experiment in which the multi-label classification and recommender system models are applied on the actual customer base of the financial services provider. In that way, it can be confirmed whether the proposed models indeed result in the expected degree of cross-sell sales. We also point out that such experiments can best be supported by appropriate marketing actions and should be planned in a sufficiently long testing period. When the experiment yields comparable results to the results of this study, it can be concluded that this study can be generalized.

However, regardless the above-mentioned limitations our study is the first to compare multi-label classification techniques and recommender approaches in the financial services sector. Hence, we consider this study a valuable contribution to literature.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2019.05.037.

References

Abbas, A., Bilal, K., Zhang, L., & Khan, S. U. (2015). A cloud based health insurance plan recommendation system: A user centered approach. *Future Generation Computer Systems, 43*, 99–109.

Adomavicius, G., & Kwon, Y. (2012). Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering, 24*, 896–911.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, 17*, 734–749.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning, 36*, 105–139.

Baumann, P., Hochbaum, D. S., & Yang, Y. T. (2019). A comparative study of the leading machine learning techniques and two new optimization algorithms. *European Journal of Operational Research, 272*(3), 1041–1057.

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., & Jones, Z. M. (2016). mlr: Machine learning in R. *Journal of Machine Learning Research, 17*, 1–5.

Bogaert, M., Ballings, M., Hosten, M., & Van den Poel, D. (2017). Identifying soccer players on Facebook through predictive analytics. *Decision Analysis, 14*, 274–297.

Bogaert, M., Ballings, M., & Van den Poel, D. (2018). Evaluating the importance of different communication types in romantic tie prediction on social media. *Annals of Operations Research, 263*, 501–527.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence* (pp. 43–52). Morgan Kaufmann Publishers Inc.

Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.

Charte, F., & Charte, D. (2015). Working with multilabel datasets in R: The mldr package. *The R Journal, 7*, 149–162.

Debaere, S., Coussement, K., & De Ruyck, T. (2018). Multi-label classification of member participation in online innovation communities. *European Journal of Operational Research, 270*, 761–774.

Demiriz, A. (2004). Enhancing product recommender systems on sparse binary data. *Data Mining and Knowledge Discovery, 9*, 147–170.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

De Weerdt, J., De Backer, M., Vanthienen, J., & Baesens, B. (2011). A robust F-measure for evaluating discovered process models. In *Proceedings of the 2011 IEEE symposium on computational intelligence and data mining (CIDM)* (pp. 148–155). IEEE.

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association, 56*, 52–64.

Felfernig, A., Isak, K., Szabo, K., & Zachar, P. (2007). The VITA financial services sales support environment. In *Proceedings of the national conference on artificial intelligence* (p. 1692). AAAI Press; MIT Press. 1999.

Felfernig, A., & Kiener, A. (2005). Knowledge-based interactive selling of financial services with FSAdvisor. In *Proceedings of the national conference on artificial intelligence* (p. 1475). AAAI Press; MIT Press. 1999.

Felfernig, A., Teppan, E., & Gula, B. (2007). Knowledge-based recommender technologies for marketing and sales. *International Journal of Pattern Recognition and Artificial Intelligence, 21*, 333–354.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics, 11*, 86–92.

Geuens, S., Coussement, K., & De Bock, K. W. (2018). A framework for configuring collaborative filtering-based recommendations derived from purchase data. *European Journal of Operational Research, 265*, 208–218.

Gonzalez-Carrasco, I., Colomo-Palacios, R., Luis Lopez-Cuadrado, J., Garcia-Crespo, A., & Ruiz-Mezcua, B. (2012). PB-ADVISOR: A private banking multi-investment portfolio advisor. *Information Sciences, 206*, 63–82.

Hahsler, M. (2011). *Developing and testing top-n recommendation algorithms for 0-1 data using recommenderlab*. NSF Industry University Cooperative Research Center for Net-Centric Software and System.

Hahsler, M., Buchta, C., Gruen, B., Hornik, K., Johnson, I., & Borgelt, C. (2019). R-package arules: Mining Association Rules and Frequent Itemsets (Version 1.6-3). Retrieved from https://CRAN.R-project.org/package=arules.

Hahsler, M., & Vereet, B. (2019). R-package recommenderlab: Lab for Developing and Testing Recommender Algorithms (Version 0.2-4). Retrieved from https://CRAN.R-project.org/package=recommenderlab.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*, 1263–1284.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS), 22*, 5–53.

Ishwaran, H., & Kogalur, U. B. (2019). R-package randomForestSRC: Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC) (Version 2.9.0). Retrieved from https://CRAN.R-project.org/package=randomForestSRC.

Kamakura, W. A., Ramaswami, S. N., & Srivastava, R. K. (1991). Applying latent trait analysis in the evaluation of prospects for cross-selling of financial services. *International Journal of Research in Marketing, 8*, 329–349.

Kamakura, W. A., Wedel, M., de Rosa, F., & Mazzon, J. A. (2003). Cross-selling through database marketing: A mixed data factor analyzer for data augmentation and prediction. *International Journal of Research in Marketing, 20*, 45–65.

Knott, A., Hayes, A., & Neslin, S. A. (2002). Next-product-to-buy models for cross-selling applications. *Journal of Interactive Marketing, 16*, 59–75.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the 1995 international joint conferences on artificial intelligence, IJCAI. Stanford, CA, pp. 1137–1145.
Kohavi, R., & Provost, F. (1998). Glossary of terms. *Machine Learning, 30*, 271–274.
Kursa, M. B. (2014). rFerns: An implementation of the random ferns method for general-purpose machine learning. *Journal of Statistical Software, 61*, 1–13.
Kursa, M. B., & Wieczorkowska, A. A. (2014). Multi-label ferns for efficient recognition of musical instruments in recordings. In *Proceedings of the international symposium on methodologies for intelligent systems* (pp. 214–223). Springer International Publishing.
Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge University Press.
Li, S., Sun, B., & Montgomery, A. L. (2011). Cross-selling the right product to the right customer at the right time. *Journal of Marketing Research, 48*, 683–700.
Li, S., Sun, B., & Wilcox, R. T. (2005). Cross-selling sequentially ordered products: An application to consumer banking services. *Journal of Marketing Research, 42*, 233–239.
Lin, W., Alvarez, S. A., & Ruiz, C. (2002). Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery, 6*, 83–105.
Liu, C., & Cai, S. (2007). Customer cross-selling model based on customer maturity and product grade. *International Management Review, 3*, 50.
Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems, 74*, 12–32.
Lü, L., Medo, M., Yeung, C. H., Zhang, Y. C., Zhang, Z. K., & Zhou, T. (2012). Recommender systems. *Physics Reports, 519*, 1–49 Review Section of Physics Letters.
Madjarov, G., Kocev, D., Gjorgjevikj, D., & Dzeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition, 45*, 3084–3104.
Mobasher, B., Dai, H., Luo, T., & Nakagawa, M. (2001). Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd international workshop on web information and data management* (pp. 9–15). ACM.
Montañes, E., Senge, R., Barranquero, J., Ramón Quevedo, J., José del Coz, J., & Hüllermeier, E. (2014). Dependent binary relevance models for multi-label classification. *Pattern Recognition, Handwriting Recognition and other PR Applications, 47*, 1494–1508.
Musto, C., Semeraro, G., Lops, P., de Gemmis, M., & Lekkas, G. (2015). Personalized finance advisory through case-based recommender systems and diversification strategies. *Decision Support Systems, 77*, 100–111.
Nemenyi, P. (1963). *Distribution-free multiple comparisons*. Princeton University Ph.D. thesis.
Nair-Benrekia, N.-Y., Kuntz, P., & Meyer, F. (2015). Learning from multi-label data with interactivity constraints: An extensive experimental study. *Expert Systems with Applications, 42*, 5723–5736.
Prinzie, A., & Van den Poel, D. (2008). Random Forests for multiclass classification: Random MultiNomial Logit. *Expert Systems with Applications, 34*, 1721–1732.
Prinzie, A., & Van den Poel, D. (2006). Incorporating sequential information into traditional classification models by using an element/position-sensitive SAM. *Decision Support Systems, 42*, 508–526.
Probst, P., Au, Q., Casalicchio, G., Stachl, C., & Bischl, B. (2017). Multilabel classification with R package mlr. arXiv preprint arXiv:1703.08991.
Pu, P., Chen, L., & Hu, R. (2011). A user-centric evaluation framework for recommender systems. In *Proceedings of the 5th ACM conference on recommender systems* (pp. 157–164). ACM.
Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2009). Classifier chains for multi-label classification. *Machine Learning and Knowledge Discovery in Databases*, 254–269.
Reinartz, W. J., & Kumar, V. (2003). The impact of customer relationship characteristics on profitable lifetime duration. *Journal of Marketing, 67*, 77–99.
Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on computer supported cooperative work* (pp. 175–186). ACM.
Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295). ACM.
Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on electronic commerce* (pp. 158–167). ACM.
Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 253–260). ACM.
Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 210–217). ACM Press/Addison-Wesley Publishing Co.
Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*, 427–437.
Thuring, F., Nielsen, J. P., Guillen, M., & Bolance, C. (2012). Selecting prospects for cross-selling financial products using multivariate credibility. *Expert Systems with Applications, 39*, 8809–8816.
Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *Int J Data Warehousing and Mining, 3*, 1–13.
Van den Poel, D. (2003). Predicting mail-order repeat buying: Which variables matter? *Tijdschrift voor Economie en Management, 48*, 371–404.
Van den Poel, D., & Lariviere, B. (2004). Customer attrition analysis for financial services using proportional hazard models. *European Journal of Operational Research, 157*, 196–217.
van Rijsbergen, C. J. (1979). *Information retrieval* (2nd ed.). London: Butterworths.
Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin, 1*, 80–83.
Zhang, M.-L., & Zhou, Z.-H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering, 26*, 1819–1837.
Zhao, Y., & Cen, Y. (2013). *Data mining applications with R*. Academic Press.
Zufferey, D., Hofer, T., Hennebert, J., Schumacher, M., Ingold, R., & Bromuri, S. (2015). Performance comparison of multi-label learning algorithms on clinical data for chronic diseases. *Computers in Biology and Medicine, 65*, 34–43.