# Hybrid Recommender Systems for Next Purchase Prediction Based on Optimal Combination Weights

Nicolas Haubner[1](✉) and Thomas Setzer[2]

[1] Institute of Information Systems and Marketing,
Karlsruhe Institute of Technology, Karlsruhe, Germany
nicolas.haubner2@partner.kit.edu
[2] Ingolstadt School of Management,
Catholic University of Eichstätt-Ingolstadt, Ingolstadt, Germany
thomas.setzer@ku.de

**Abstract.** Recommender systems (RS) play a key role in e-commerce by pre-selecting presumably interesting products for customers. Hybrid RSs using a weighted average of individual RSs' predictions have been widely adopted for improving accuracy and robustness over individual RSs. While for regression tasks, approaches to estimate optimal weighting schemes based on individual RSs' out-of-sample errors exist, there is scant literature in classification settings. Class prediction is important for RSs in e-commerce, as here item purchases are to be predicted. We propose a method for estimating weighting schemes to combine classifying RSs based on the variance-covariance structures of the errors of individual models' probability scores. We evaluate the approach on a large real-world e-commerce data set from a European telecommunications provider, where it shows superior accuracy compared to the best individual model as well as a weighting scheme that averages the predictions using equal weights.

**Keywords:** Hybrid recommender systems · Forecast combination · Optimal weights · Demographic filtering

## 1 Introduction

Personalized information systems (IS) are crucial nowadays in the areas of marketing and sales, providing a unique experience to users with the help of dialogues and relevant content. Advances in technology have made it possible to collect and process increasing amounts of data, such as customer profiles, activities and interests. Turning this data into actionable insights is not only key to acquiring and retaining customers, but also to providing suitable purchasing recommendations for up- and cross-selling items relevant to and appreciated by existing customers in order to increase customer lifetime values.

In this spirit, recommender systems (RS) are personalized ISs with the goal of helping customers make better (purchasing) decisions. There are different criteria for measuring the quality of an RS, e.g. serendipity, diversity, and predictive accuracy. In this paper, we


F. Ahlemann et al. (Eds.): WI 2021, LNISO 47, pp. 56–71, 2021.
https://doi.org/10.1007/978-3-030-86797-3_4

focus on the latter aspect. IS research has shown that the accuracy of recommendations, i.e. the perceived personalization, is of key importance for customers to adopt an RS as a decision aid, and thus, purchasing recommended items. More accurate RSs increase decision quality and also help companies retain customers [1].

Increasing the predictive accuracy of an RS can be achieved in several ways, e.g. by applying an improved predictive algorithm, tuning hyper-parameters or collecting additional input data for single RS techniques used. In addition to and independent of the former approaches, accuracy and robustness of RSs can be improved by combining multiple different prediction algorithms. This is called a hybrid RS (HRS). HRSs have been shown in the IS literature to improve decision quality and satisfaction with the system, compared to using only single recommendation methods such as collaborative or content-based approaches [2].

There are different ways of combining RSs into a hybrid, e.g. weighted, switching, mixed, or feature combination. In this study, we focus on weighted HRSs. The literature on how to select combination weights in weighted HRSs is very limited, specifically in the context of purchase predictions. Providing products or product categories of interest to a current user is key to content and affiliate marketing, generating leads and developing existing customers in terms of up- and cross-selling endeavors.

In [3], the authors propose a method of estimating optimal weights (OW) for combining multiple RSs in a rating prediction scenario. Their approach derives in-sample OW that minimize the mean squared error (MSE) of the HRS on the training data given certain assumptions. We transfer the weighting method from regression to a multi-category classification problem, where the goal is to predict the next purchase of a given customer based on the customer profile. For that, we use the Brier score, which quantifies the mean squared deviation of the estimated purchase probability from the true outcome. The Brier score is therefore analogous to the MSE in regression settings and is used in this work to estimate weights for combining probability scores of multiple classifiers.

The approach is evaluated on a labeled real-world data set from a large European telecommunications provider, where it is used to predict purchasing probabilities for three categories of mobile devices. The task is to predict the conditional probability, given that a certain customer is going to buy a mobile device, in which category it will be. Thus, the problem can be regarded as a top-1 recommendation task. Experimental results show that the proposed classifier weighting method leads to significant improvements, both in the Brier score and the accuracy score, compared to both the individual models as well as a combination where all models receive equal weights.

The remainder of this paper is organized as follows. Section 2 provides foundations of HRSs and forecast combination. Section 3 describes the proposed classifier weighting method. Section 4 outlines the experimental design to evaluate the proposed method on a real-world data set. Section 5 reports the experimental results, and Sect. 6 discusses the benefits and shortcomings of the proposed approach. Finally, we conclude and suggest directions for future research in Sect. 7.

## 2 Related Work

In this section, we review foundations of the proposed weighting approach. Section 2.1 gives an overview of HRSs with a focus on weighted approaches. Section 2.2 provides

background on statistical forecast combination. Section 2.3 summarizes the research gap and motivates the novel method.

### 2.1 Hybrid Recommender Systems

An RS is a software system designed for estimating users' interest for products, based on their past purchases and possibly other inputs, and suggesting them those items with the highest estimated interest. RSs reduce information overload and improve users' decision quality by limiting the number of options. For companies, they increase sales and help market long-tail items which would otherwise be hard to find. RSs are nowadays used by, among others, e-commerce sites, digital marketing systems, social networks, and streaming platforms, where their advantages have been shown extensively [2].

An RS's quality relates to criteria such as serendipity, diversity, and accuracy. Serendipity denotes the ability of an RS to suggest items that a given user was not aware of, but finds interesting. Diversity refers to the composition of recommendations. Instead of suggesting several similar items, a good RS should be able to cater to the different interests of a given user. Finally, an accurate RS makes recommendations which fit user needs, such that the products are then taken by users with high probability, e.g. a customer ultimately purchases suggested products or watches suggested movies (e.g. [4]).

There are several methods for calculating prediction scores from available data, such as collaborative filtering, content-based filtering, demographic filtering, or knowledge-based systems, each using different input data sources and applying different algorithms. Each RS algorithm has certain shortcomings, e.g. the cold-start problem, where collaborative filtering methods are not able to provide recommendations for new users or new items (e.g. [4]).

HRSs combine two or more individual RSs in order to alleviate those problems as well as improve accuracy and robustness. Burke [5] classifies HRSs into seven types: weighted, switching, mixed, feature combination, cascade, feature augmentation, and meta-level. In this study, we focus on weighted HRSs, where several individual RSs calculate predictions independently, and those predictions are then combined using an aggregation function. While it has been shown that using a weighted average of RSs' predictions often leads to increased accuracy due to reduced model variance, published work on the selection of combination weights is scarce. In [6], different supervised models like ridge regression, neural networks, or gradient boosted decision trees for learning weights are compared. In [3], a model to learn weighting schemes from the errors observed for individual models is transferred from the forecasting to the RS domain, using the error covariance structure of the RSs to estimate OW. The model transferred is the one introduced in [7], which will be described in more detail in the next section.

### 2.2 Statistical Forecast Combination

In statistical forecasting, the combination of multiple prediction models has been subject to a large body of research. In [7], a weighting strategy is introduced which, for two combined models, can be shown to minimize the MSE in-sample, given the individual

forecasts are unbiased, i.e. they do not consistently over- or underestimate the true values, and the performance of the individual forecasts is time-invariant. This weighting strategy is coined OW.

OW can generally be calculated for $k$ prediction models (see e.g. [8]): let $y$ be the vector of actual outcomes and $\hat{y}_l$ model $l$'s predictions for the entries in $y$. Assuming error vectors $e_l = y - \hat{y}_l$, $l \in \{1, \ldots, k\}$ of the individual models are multivariate normal with mean 0, OW can be learned that minimize the MSE over available ratings in $y$. With $\Sigma_E$ denoting the variance-covariance matrix of the error matrix $E = (e_1, \ldots, e_k)$, and $\vec{1}$ as a $k$-dimensional column vector with all ones, Eq. (1) derives the OW vector.

$$w = \frac{\Sigma_E^{-1}\vec{1}}{\vec{1}'\Sigma_E^{-1}\vec{1}} \tag{1}$$

Note that Eq. (1) minimizes the sum of squared deviations from zero (as of the unbiasedness assumption) subject to the constraint that the weights sum up to one, i.e. a weighted average. Although optimal in-sample, OW has often been reported to be outperformed on unseen data by more robust weighting strategies such as giving equal weights to all forecasts, i.e. a simple average (SA) (e.g. [9, 10]). This observation is called the "forecast combination puzzle". It can be explained by the fact that learned weights like OW must be estimated from past errors, often with rather small data sets available. Hence, OW can overfit the training data due to high model variance. SA, on the other hand, has no variance as it does not adjust weights to training data and is therefore more robust (e.g. [10]).

Contrary to the forecast combination puzzle, in [3] it is shown that given sufficient amounts of training observations, OW can be learned that are close to the ex-post OW (i.e. the unknown linear weight vector leading to the smallest out-of-sample MSE). The authors analyze this approach on a large publicly available data set with ratings of movies and find that it leads to accuracy improvements over the best individual RS as well as SA.

### 2.3 Contribution of this Paper

In summary, little research has been published on the selection of combination weights in weighted HRSs. As described above, there exist some weighting strategies for regression scenarios, mainly rating prediction, but for classification tasks, binary or multi-class, we are not aware of analytical methods for weighting different kinds of algorithms.

However, those kinds of problems appear very often in e-commerce, where a company wants to estimate, for a given user, purchasing probabilities of different products or product categories in order to show personalized advertisements or select suitable customers for marketing campaigns. In this paper, we propose an analytical weighting procedure to increase the accuracy and robustness of a multi-class classifier ensemble over the best individual classifier as well as SA. The proposed technique offers a means to increase accuracy and robustness without requiring expensive brute-force search or additional input data.

Commonly, e-commerce companies already test and compare different algorithms with the goal of maximizing predictive accuracy. Depending on the size of a company and

its number of customers, an accuracy increase as small as 1% can lead to a significantly higher profit. The proposed approach offers a simple and efficient means to combine their existing methods and thus achieve higher levels of performance and profit.

We adapt the method introduced in [3] of learning combination weights for combining multiple RSs in a rating prediction task. We combine classifying RSs based on the covariance structures of the individual models' probability scores such that the Brier score is minimized in the same fashion as the MSE is minimized in regression settings. Next purchase (class) predictions on unseen data are then derived as the class with the highest probability score.

As described in Sect. 1, both the importance of accurate RSs and the benefits of HRSs have been demonstrated in IS research. The weighting method proposed in this paper therefore provides a relevant contribution both to the existing body of research and to practitioners, mainly large companies with substantial data available and many customers.

## 3 Methodology

This section introduces the approach to estimate OW for combining predictions of classification algorithms. Section 3.1 considers the assumptions and requirements of the weighting method. Section 3.2 describes the estimation of optimal combination weights in detail.

### 3.1 Model Assumptions

The classifier weighting is based on statistical forecast combination, as introduced in Sect. 2.2. In regression settings, Eq. (1) ensures a minimal in-sample MSE given that the individual models' errors follow a multivariate normal distribution with mean 0, i.e. the models are unbiased.

Our adapted classifier weighting scheme relies on the Brier score [11] as the classification equivalent of the MSE. For $c$ possible outcomes (classes) and $n$ observations, it calculates as shown in Eq. (2), where $y_{ij}$ represents the actual outcome for observation $i$ and class $j$, which is either 0 or 1, and $\hat{y}_{ij}$ represents the estimated probability with $0 \leq \hat{y}_{ij} \leq 1$ and $\sum_{j=1}^{c} \hat{y}_{ij} = 1, i \in \{1, \ldots, n\}$.

$$BS = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} \left(y_{ij} - \hat{y}_{ij}\right)^2 \tag{2}$$

For each observation and each class label, we calculate the deviation between the predicted probability of the observation pertaining to the class and the true outcome. For each observation, the predicted probabilities sum to 1, and the true outcome is 1 for one class label and 0 for all the other labels. Since the errors are flattened, yielding error vectors of length $nc$, the deviations between prediction and ground truth sum exactly to 0 for each observation. Consequently, the mean deviation for each flattened error vector is also 0. Regarding the multivariate normality of the error vectors, respective analyses of the data set are provided in Sect. 5.1.

Another assumption of the classifier weighting method is that the minimization of the Brier score of a classifier ensemble results in an accuracy gain over all individual classifiers as well as an equal weights combination. We expect the Brier score to be an appropriate metric due to its interpretation as the MSE in probability estimation.

### 3.2 Classifier Weighting Method

Input to the method is a labeled classification data set with $n$ observations, and $k$ classification models. The output is $\widehat{\boldsymbol{w}}$, the estimate of the out-of-sample OW vector with $\widehat{\boldsymbol{w}} \in \mathbb{R}^k$ and $\sum_{l=1}^{k} \widehat{w}_l = 1$. The number of classes in the data set is denoted by $c$. A portion of the input data is held out, resulting in two subsets, the training set with $n_t$ observations and the holdout set with $n_h$ observations. The split is performed stratified, i.e. the class distributions in the training and holdout set are practically equal.

| Instance | Class | $\hat{y}_1$ | $\hat{y}_2$ | y |
|---|---|---|---|---|
| 1 | 1 | 0.3343 | 0.2531 | 0 |
| | 2 | 0.1396 | 0.3511 | 0 |
| | 3 | 0.5261 | 0.3958 | 1 |
| 2 | 1 | 0.0192 | 0.0982 | 0 |
| | 2 | 0.4492 | 0.4895 | 1 |
| | 3 | 0.5316 | 0.4123 | 0 |
| 3 | 1 | 0.2614 | 0.1163 | 0 |
| | 2 | 0.4296 | 0.4690 | 1 |
| | 3 | 0.3091 | 0.4148 | 0 |

**Fig. 1.** Example for prediction vectors and actual outcomes with $n = 3$ instances, $c = 3$ classes, and $k = 2$ classification models

All classifiers are fitted on the training set. Each classifier $l \in \{1, \ldots, k\}$ then makes probability predictions $\widehat{Y}_{hl} \in [0, 1]^{n_h \times c}$ on the holdout set. These predictions are flattened into a prediction vector $\hat{\boldsymbol{y}}_{hl}$ of length $n_h c$, which contains predicted probabilities for each instance $i \in \{1, \ldots, n_h\}$ and for each class $j \in \{1, \ldots, c\}$. $\boldsymbol{y}_h = (y_{1,1}, \ldots, y_{n_h,c})'$ denotes the vector of true outcomes in the holdout set. For each instance, $\boldsymbol{y}_h$ contains 1 for the actual class label of the instance, and 0 for all other class labels. Figure 1 shows an example of two prediction vectors and the true outcome. For simplicity, we omit the $h$ subscript in this and the following figures.

For each classifier $l$, the vector $\hat{\boldsymbol{y}}_{hl}$ of predicted probabilities is then compared to the vector $\boldsymbol{y}_h$ of actual outcomes. The error vector for classifier $l$ is calculated as $e_{hl} = y_h - \hat{y}_{hl}$. For each of the $k$ classifiers, this error vector is computed, yielding an error matrix $E_h = (e_{h1}, \ldots, e_{hk})$. Calculating OW from those error vectors can be shown to minimize the Brier score in-sample, analogous to the MSE in a regression setting. Figure 2 displays the error matrix for the predictions from Fig. 1.

With $\Sigma_h$ as the variance-covariance matrix of $E_h$, the OW estimate $\widehat{\boldsymbol{w}}$ can be computed using Eq. (1). The variance-covariance matrix of the error matrix from Fig. 2 is given by (Eq. 3)

$$\Sigma_h = \begin{pmatrix} 0.1789 & 0.1730 \\ 0.1730 & 0.1825 \end{pmatrix}. \tag{3}$$

| $\mathbf{e}_1$ | $\mathbf{e}_2$ |
|---|---|
| -0.3343 | -0.2531 |
| -0.1396 | -0.3511 |
| 0.4739 | 0.6042 |
| -0.0192 | -0.0982 |
| 0.5508 | 0.5105 |
| -0.5316 | -0.4123 |
| -0.2614 | -0.1163 |
| 0.5704 | 0.5310 |
| -0.3091 | -0.4148 |

**Fig. 2.** Example for error matrix based on the predictions in Fig. 1

The weight vector estimated in this example is $\widehat{\boldsymbol{w}} = (0.6163, 0.3837)'$. Finally, training and holdout set are concatenated again, and all $k$ individual classifiers are re-fitted on all $n$ observations. This is to ensure that all models can process as many observations as possible in the training phase. The classifiers' probability predictions on new, unseen data are subsequently combined using the weight vector $\widehat{\boldsymbol{w}}$ estimated on the training set.

Many classification algorithms yield class membership scores that can be used to rank observations based on their likelihood to pertain to a certain class. However, those scores in general cannot be interpreted as proper probability estimates, since they are not well-calibrated, i.e. predicted class membership scores do not match ex-post probabilities [12]. While this is not an issue for class predictions of single classifiers, in classifier ensembles it is important to have reliable probability estimates. Therefore, we compare the OW estimation with and without calibration. For the calibration setting, we use isotonic regression as introduced in [13].

## 4 Experimental Design

We now describe the experiments conducted to evaluate the proposed weighting approach. Section 4.1 describes the use case and data set used for evaluation. Section 4.2 introduces the individual classifier methods used for the HRS. In Sect. 4.3, details about the experiments and evaluation criteria are given.

### 4.1 Use Case and Data Set

For the evaluation of the proposed classifier weighting scheme, we used a proprietary real-world data set from a large European telecommunications provider. Figure 3 displays the schema of the data set. It contains several hundred thousand purchases of mobile devices by customers. All purchases occurred in the years 2018 and 2019. In the figure, the last column represents the target variable, the first two columns are metadata for identification, and the columns in between are predictors.

| Order date | Customer ID | Socio-demographic variables | Contract properties | Mobile data usage | ... | Device type bought (class) |
|---|---|---|---|---|---|---|
| 1/1/2018 | ... | ... | ... | ... | ... | 1 |
| 1/1/2018 | ... | ... | ... | ... | ... | 2 |
| ... | ... | ... | ... | ... | ... | ... |
| 12/31/2019 | ... | ... | ... | ... | ... | 3 |

**Fig. 3.** Schematic display of the data set used for evaluation

The mobile devices are divided into three categories. The goal is to predict, for each of the $c = 3$ categories, the conditional probability that the given customer will select the respective category, given a purchase. The category with the highest estimated probability is then recommended. The most frequent of the three class labels occurs in 48% of cases in the data set. Thus, a simple classifier which always predicts that label would already achieve an accuracy score of 48%, which can serve as a lowest bound for more sophisticated models.

The data set contains more than 40 predictor variables, consisting of customer properties such as sociodemographics, characteristics of the customer's contract, and aggregated behavioral information such as mobile data usage. The data types of the predictors are mixed, comprising binary, integer, real-valued as well as categorical variables. All values of the predictors were measured immediately before the respective purchase, representing a snapshot of the respective customer and contract in order to recognize purchasing patterns.

### 4.2 Individual Classifiers

This section describes the individual models used to test the weighting method. Since the model's inputs are vectors representing customers via their respective properties, the method used here can be classified as demographic filtering (e.g. [14]), although the predictors do not only contain demographic information. In total, $k = 7$ classifying algorithms were combined, which are briefly outlined here. We used the implementations in the Python package *scikit-learn* [15] for the individual classifiers.

- **Logistic regression**: The logistic regression model assumes a linear relationship between the predictor variables and the log-odds of the positive outcome of a binary

dependent variable (e.g. [16]). The model can be extended for non-binary classification either fitting a one-vs-all model for each class label or minimizing the multinomial logistic loss. The latter is used here.

- ***k*-nearest neighbors classifier**[1]: A $k$-nearest neighbors classifier predicts, for a given instance to classify, the class which most occurs in the $k$ training points with the smallest distance to that instance [17]. For probabilistic predictions, the class distribution of those $k$ instances is predicted. The distance metric used here is the Euclidean distance, and the number of neighbors considered was set to $k = 5$.
- **Multi-layer perceptron**: A multi-layer perceptron is a frequently-used form of neural networks, consisting of an input layer with $m$ nodes (the number of features), an output layer with $c$ nodes (the number of classes), and one or more hidden layers (e.g. [16]). The nodes of the hidden layer use a nonlinear activation function, in our case the rectified linear unit $f(x) = \max\{0, \text{x}\}$. The weights between nodes are initialized randomly and then sequentially updated using the back propagation algorithm, which computes the gradient of the loss function with respect to each weight. The weight optimization is done using the efficient stochastic gradient descent method Adam [18], and the maximum number of iterations is set to 1000.
- **Decision tree**: The decision tree algorithm [19] learns simple "if-else" style decision rules by recursively splitting the data set with respect to a certain variable and value in order to create subsets which are more pure in terms of class distribution. We used a maximum depth of 5 in order not to overfit the training set.
- **Random forest**: The random forest algorithm [20] fits an ensemble of decision trees on the training data. By randomly selecting bootstrap samples of data and randomly selecting a subset of variables available for splitting at each node, the trees in the ensemble are partially independent, reducing the model variance and thus alleviating a single decision tree's tendency to overfit the training data. For predicting probabilities on new data, the average of predicted probabilities of all trees in the ensemble is calculated. We chose a number of 100 trees with a maximum depth of 5 for the forest.
- **AdaBoost**: AdaBoost [21] is an ensemble method which fits simple base learners sequentially, where in each iteration, weights for previously misclassified instances are increased such that the next base learner is forced to focus on more difficult cases. For prediction, the outputs of all base learners are aggregated. We used 50 decision trees with a depth of 1, also known as "decision stumps".
- **Gradient boosting**: Gradient boosting [22] sequentially builds an additive model. In each iteration, $c$ (the number of classes) regression trees are fitted on the negative gradient of the loss function, which for probabilistic outputs is the deviance. We chose a value of 100 iterations.

### 4.3 Evaluation and Benchmarks

As mentioned in Sect. 4.1, the prediction task in this business case was to recommend one of $c = 3$ possible classes of mobile devices to each customer in the test set. Therefore, an ensemble of three-class classifiers was used. In order to evaluate the classifier weighting approach we propose, the following methods were compared:

[1] Note that in this bullet point only, $k$ represents the number of neighbors. In the rest of the article, $k$ is used to denote the number of classifiers combined in the HRS.

- **Individual classifiers**: For each of the seven models described in Sect. 4.2, the individual performance on the test set was calculated.
- **SA**: An equal weights average of the predictions of all seven classifiers on the test set was used as a benchmark for the hybrid approach.
- **OW estimate**: This is the approach proposed in this paper (see Sect. 3 for details).
- **Out-of-sample OW**: The linear weight vector with ex-post minimal Brier score, calculated on the test set, serves as an upper performance bound for any linear weight vector. The goal of the OW estimate is to come as close as possible to this performance.

For each of the mentioned methods, there are two treatments: first, the probabilistic predictions are taken as-is. Second, the predicted probabilities are calibrated using isotonic regression before making predictions or combining the probabilistic predictions. The un-calibrated and calibrated treatments are compared.

In order to evaluate the weighting approach and compare it to other strategies, 10% of the data set was used as a test set. Another 10% of the data set was used as a holdout set to calculate out-of-sample errors of the individual classifiers in order to estimate OW, as described in Sect. 3.2. This leaves 80% of the data set as a training set.

We used two evaluation metrics in the experiment: the accuracy score and the Brier score which was also used for weight estimation. Those two metrics were chosen because the proposed weighting approach aims at increasing the accuracy of a classifier ensemble over all individual components as well as an SA combination by minimizing the Brier score. As mentioned in Sect. 3.1, we expect the minimization of the Brier score to result in a significant accuracy gain.

For reasons of robustness, the experiment was repeated ten times with random training-holdout-test allocations. Accuracy and Brier scores were averaged over those runs, and their standard deviations are reported.

## 5 Empirical Evaluation

This section contains the experimental results of comparing the proposed classifier weighting method to the afore-mentioned benchmarks. Section 5.1 describes the data preparation, i.e. checking the model requirements. Section 5.2 reports the results.

### 5.1 Data Preparation

As mentioned above, the weighting method requires unbiased individual estimators (mean errors of 0) and multivariate normal error vectors. We described in Sect. 3.1 that the mean errors are 0 when using the flattened deviation between predicted probabilities and true outcomes as error vectors. Now, we inspect the distribution of deviations.

Figure 4 displays, for each individual classifier, a histogram of out-of-sample errors, using ten equal-width bins. The vertical axes are not labeled since the number of observations in the test set would give away the number of observations in the entire data set (see Sect. 4.1). The left part of the figure shows the histograms when feeding error vectors into the weight estimation as-is, i.e. without calibration. It is clear to see that the errors are not normally distributed. Some of the classifiers partially exhibit a bell-shaped

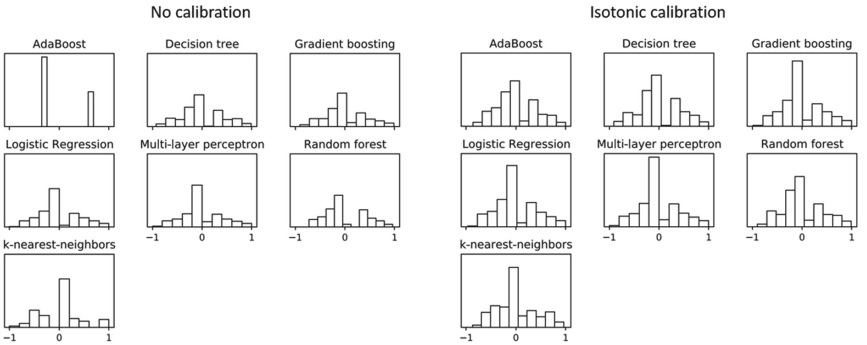


**Fig. 4.** Histograms of individual classifiers' out-of-sample errors without calibration (left) and with isotonic calibration (right)

distribution, however all with a gap around 0. Others, especially AdaBoost, are nowhere near bell-shaped.

However, applying the mentioned classifier calibration technique using isotonic regression changes the error distribution. The right part of Fig. 4 shows the same plot, but this time after isotonic calibration of each classifier. While there is still a gap near 0, all distributions now exhibit a bell-shaped form. They are still not normally distributed, but the calibration helps to better approach the assumption.

### 5.2 Results

Table 1 displays the results of comparing the HRS using OW estimation, an HRS using SA combination, and the individual classifiers. Both for the accuracy and the Brier score, the mean and standard deviation over ten runs are reported. For better comparability, the percentage differences between the OW estimation and the other methods is also reported for both metrics (columns "Diff."). As mentioned in Sect. 4.3, in addition to the individual classifiers, the SA combination and the OW estimate, the results using out-of-sample OW are also reported (last row) as an upper bound for the performance of a linear weighting vector.

The table shows that the combination using OW estimation clearly outperforms all individual methods as well as the SA combination. The best-performing individual classifier, which is the neural network with an accuracy of 66.8%, has a 2.4% lower accuracy and a 4.99% higher Brier score than the HRS using the OW estimate. An SA combination using equal weights slightly outperforms the best individual method, but leads to a 2.2% lower accuracy and a 6.25% higher Brier score than estimated OW.

The results also show that the estimated OW vector is very close in both accuracy (0.07% lower) and Brier score (0.07% higher) to the ex-post, out-of-sample OW vector. This indicates that the weighting approach proposed in this study can yield weight vectors that are close to the best possible linear weighting HRS.

Table 2 displays the results when all classifiers' probability estimates are calibrated using isotonic regression before estimating OW. After calibration, all classifiers' Brier

**Table 1.** Comparison of performance between a hybrid recommender system using optimal weight estimation, a simple average combination, and all individual classifiers

| Method | Accuracy (std.) | Diff | Brier score (std.) | Diff |
|---|---|---|---|---|
| Logistic regression | 0.6440 (0.0012) | +6.14% | 0.1595 (0.0004) | −10.70% |
| $k$-nearest-neighbors | 0.6280 (0.0016) | +8.84% | 0.1677 (0.0004) | −15.09% |
| Multi-layer perceptron | 0.6675 (0.0015) | +2.40% | 0.1499 (0.0005) | −4.99% |
| Decision tree | 0.6463 (0.0013) | +5.76% | 0.1600 (0.0003) | −10.98% |
| Random forest | 0.6430 (0.0014) | +6.30% | 0.1649 (0.0004) | −13.63% |
| AdaBoost | 0.6503 (0.0012) | +5.10% | 0.2187 (0.0000) | −34.88% |
| Gradient boosting | 0.6590 (0.0013) | +3.71% | 0.1529 (0.0003) | −6.88% |
| Simple average | 0.6688 (0.0012) | +2.20% | 0.1519 (0.0002) | −6.25% |
| Optimal weight estimate | 0.6835 (0.0013) | | 0.1424 (0.0003) | |
| Ex-post optimal weights | 0.6840 (0.0013) | −0.07% | 0.1423 (0.0003) | +0.07% |

scores are in a range between 0.15 and 0.16. Their individual accuracies are not significantly affected, with one exception: the calibrated nearest-neighbors classifier has an accuracy of 64.8%, as compared to 62.8% for the non-calibrated version. This indicates that the calibration changed the order of the class ranking for some instances, leading to a higher number of correct class predictions.

**Table 2.** Results analogous to Table 1 after isotonic calibration of individual classifiers

| Method | Accuracy (std.) | Diff | Brier score (std.) | Diff |
|---|---|---|---|---|
| Logistic regression | 0.6440 (0.0014) | +6.40% | 0.1593 (0.0004) | −10.83% |
| $k$-nearest-neighbors | 0.6484 (0.0017) | +5.69% | 0.1561 (0.0003) | −9.01% |
| Multi-layer perceptron | 0.6673 (0.0015) | +2.68% | 0.1500 (0.0005) | −5.29% |
| Decision tree | 0.6460 (0.0013) | +6.07% | 0.1600 (0.0003) | −11.21% |
| Random forest | 0.6461 (0.0018) | +6.05% | 0.1593 (0.0003) | −10.80% |
| AdaBoost | 0.6487 (0.0011) | +5.63% | 0.1579 (0.0003) | −10.03% |
| Gradient boosting | 0.6590 (0.0012) | +3.97% | 0.1528 (0.0003) | −7.03% |
| Simple average | 0.6648 (0.0015) | +3.07% | 0.1500 (0.0002) | −5.29% |
| Optimal weight estimate | 0.6852 (0.0013) | | 0.1421 (0.0003) | |
| Ex-post optimal weights | 0.6861 (0.0013) | −0.12% | 0.1419 (0.0003) | +0.12% |

As for the weighted HRSs, the SA combination of calibrated classifiers has a slightly better (smaller) Brier score and worse (smaller) accuracy than the non-calibrated SA combination. Now, the best individual classifier, which is again the neural network, slightly outperforms the SA combination in terms of accuracy. For the estimated OW,

the Brier score is virtually unchanged, while the accuracy slightly increases when using calibration. This is probably caused by the nearest-neighbors classifier's gain in accuracy. The performance increase of the OW estimation over the best classifier (2.68%) as well as the SA (3.07%) is even higher than in the no-calibration treatment.

## 6 Discussion

In this section, we discuss and interpret the results obtained from the experimental evaluation in the previous section.

First, as apparent in Table 1, the technique of estimating OW using a subset of the available training data and then applying the learned weighting to the full data set clearly outperforms the SA combination as well as all individual methods. Although seven classification models were combined, meaning a weight vector of length $k = 7$ had to be estimated, the estimated OW comes very close to the ex-post OW, both in terms of accuracy and Brier score. This is probably due to the rather large data set used in the experiments, leading to robust weight estimates. Large e-commerce vendors usually have large data sets available, making the proposed approach a feasible and effective means to boost predictive accuracy.

In the use case of this paper, the proposed classifier weighting approach was able to increase the accuracy over the best individual classifier as well as SA by more than 2%. The impact of such an improvement depends on the business case at hand. For the project partner that provided the data set, this improvement is significant. Due to the high number of customers, being able to predict the right purchase in 2% more of the cases can lead to a considerable profit enhancement. Other large corporations, especially in e-commerce, could benefit in a similar way. On the other hand, for smaller companies with fewer customers as well as smaller data sets, other factors are more important, such as the model interpretability.

The theoretical model requirements were not entirely fulfilled in the use case, since the classifier error vectors were not normally distributed. Real-world use cases often differ substantially from theoretical requirements, which is why many approaches do not work well under those circumstances. However, the proposed approach was still able to reach an accuracy and Brier score very close to the ex-post best possible, and to improve performance over individual classifiers and SA combination. This shows that the classifier weighting is well suited for practitioners, even if the data is messy, as it often is in practice.

The classifier weighting method can be integrated into existing machine learning pipelines rather easily. Due to its analytical nature, it does not require expensive computations, and the weights are readjusted automatically without regular human intervention. Therefore, the cost-benefit ratio calculates favorably. The potential of gaining significant performance was shown in this study, and because of the added robustness, there is minimal risk of losing accuracy given sufficient data.

Second, as can be seen when comparing Table 1 to Table 2, although calibrating the probability estimates of the individual classifiers using isotonic regression led to decreasing Brier scores of the individual models (especially AdaBoost), it did not lead to significant differences in the accuracy of the OW combination. This is probably because almost all classifiers already had mostly well-calibrated scores.

Finally, while in general, a minimal Brier score does not automatically lead to the highest accuracy, in our case, the methods with the lowest Brier score (neural network for the individual methods and OW estimate for the HRSs) did have the highest accuracies as well. Especially the OW estimate, which aims at minimizing the Brier score of an HRS, outperforms all other models by more than 2% in terms of accuracy. This indicates that selecting combination weights based on the Brier score is a good strategy for creating accurate HRSs and therefore confirms our last assumption from Sect. 3.1.

## 7 Conclusion

In this paper, we presented an approach to estimate optimal combination weights for HRSs in a classification context, e.g. for purchase prediction. The weighting method fits all individual classifiers on a subset of the available training data and calculates out-of-sample errors of probabilistic predictions on the rest of the training data. Using the variance-covariance matrix of those out-of-sample errors, a weight vector is calculated which is optimal on the holdout set. Then, all classifiers are re-fitted on the entire available data set, and the calculated weight vector is used for combining predictions on new, unseen data.

Results on a real-world e-commerce data set show that this approach significantly outperforms both an SA combination, assigning equal weights to all components, as well as all individual classifiers. This is an encouraging finding, indicating that OW estimated using the Brier score is an adequate and simple method for increasing accuracy and robustness of classifiers.

This study contributes to research and practice. First, a novel and accurate analytical weighting scheme for classifiers is proposed. It contributes to the literature on weighted HRSs as well as classifier ensembles in general. For practitioners, especially companies with many customers and large data sets as well as different classifying models in use, the method provides a computationally efficient means of increasing accuracy and robustness, and thus revenue and profit, without requiring great effort to set up or maintain.

As a limiting factor, we did not engage in extensive hyper-parameter tuning for the individual classification algorithms, since the goal of this study was to demonstrate the improvement of a weighted HRS using an OW estimate over all individual models as well as an SA combination. In addition, we did not perform any feature selection or engineering. Performing both of those tasks might have improved the accuracy of the HRS even further.

Future research should investigate how well the OW estimation based on the Brier score performs in settings with fewer training observations or more classifiers. We expect that for smaller data sets, the advantage over SA decreases due to the bias-variance trade-off between those two weighting methods. Shrinking the estimated OW vector toward SA (e.g. [23]) or similar robust weighting strategies with lower variance might be a remedy against this effect.

Another interesting direction for future studies is to test the approach introduced in this paper using other algorithms, e.g. implicit feedback collaborative filtering methods (which would require other data sets), and study whether it is also able to improve an

ensemble of RSs in terms of other metrics, such as ranking metrics which are often relevant in a top-$N$ recommendation setting.

## References


1. Komiak, S.Y., Benbasat, I.: The effects of personalization and familiarity on trust and adoption of recommendation agents. MIS Q. **30**(4), 941–960 (2006)
2. Xiao, B., Benbasat, I.: E-commerce product recommendation agents: use, characteristics, and impact. MIS Q. **31**, 137–209 (2007)
3. Haubner, N., Setzer, T.: Applying optimal weight combination in hybrid recommender systems. In: Proceedings of the 53rd Hawaii International Conference on System Sciences (2020)
4. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**, 734–749 (2005). https://doi.org/10.1109/TKDE.2005.99
5. Burke, R.: Hybrid recommender systems: survey and experiments. User Model User-Adapt. Interact. **12**, 331–370 (2002). https://doi.org/10.1023/A:1021240730564
6. Jahrer, M., Töscher, A., Legenstein, R.: Combining predictions for accurate recommender systems. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 693–702. ACM, New York (2010). https://doi.org/10.1145/1835804.1835893
7. Bates, J.M., Granger, C.W.J.: The combination of forecasts. OR **20**, 451 (1969). https://doi.org/10.2307/3008764
8. Timmermann, A.: Chapter 4 forecast combinations. In: Elliott, G., Granger, C.W.J., Timmermann, A. (eds.) Handbook of Economic Forecasting, pp. 135–196. Elsevier (2006). https://doi.org/10.1016/S1574-0706(05)01004-9
9. Clemen, R.T.: Combining forecasts: a review and annotated bibliography. Int. J. Forecast. **5**, 559–583 (1989). https://doi.org/10.1016/0169-2070(89)90012-5
10. Smith, J., Wallis, K.F.: A simple explanation of the forecast combination puzzle. Oxford Bull. Econ. Stat. **71**, 331–355 (2009). https://doi.org/10.1111/j.1468-0084.2008.00541.x
11. Brier, G.W.: Verification of forecasts expressed in terms of probability. Mon. Weather Rev. **78**, 1–3 (1950)
12. Niculescu-Mizil, A., Caruana, R.: Predicting good probabilities with supervised learning. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 625–632 (2005)
13. Zadrozny, B., Elkan, C.: Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 694–699 (2002)
14. Pazzani, M.J.: A framework for collaborative, content-based and demographic filtering. Artif. Intell. Rev. **13**, 393–408 (1999). https://doi.org/10.1023/A:1006544522159
15. Pedregosa, F., et al.: Scikit-learn: Machine Learning in Python [cs]. arXiv:1201.0490 (2012)
16. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer New York (2009). https://doi.org/10.1007/b94608
17. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**, 175–185 (1992). https://doi.org/10.1080/00031305.1992.10475879
18. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
19. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. CRC press (1984)

20. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
21. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory, pp. 23–37. Springer (1995). https://doi.org/10.1007/3-540-59119-2_166
22. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**(5), 1189–1232 (2001)
23. Blanc, S.M., Setzer, T.: When to choose the simple average in forecast combination. J. Bus. Res. **69**, 3951–3962 (2016). https://doi.org/10.1016/j.jbusres.2016.05.013