

# Fancy Title

Business Analytics and Data Science Group Project

submitted to

**First Prof**  
**Second Prof**

Humboldt-Universität zu Berlin  
School of Business and Economics

Chair

by

**Claudia Guenther, Phi Nguyen, Julian Winkel**  
Immatriculation Numbers



Anything else we want to say

Berlin, Date

## Abstract

Insert abstract here

## List of Tables

1	Baseline Models . . . . .	5
2	Constructed features ordered by category (excerpt) . . . . .	6
3	Accuracy comparison . . . . .	12
4	Log-loss error comparison . . . . .	12
5	Total cost comparison . . . . .	12
6	Neural Network Hyperparameter Tuning . . . . .	13
7	Random Forest Hyperparameter Tuning . . . . .	13
8	Gradient Boost Hyperparameter Tuning . . . . .	13
9	Cost Improvement from Ensembling . . . . .	15

## List of Figures

1	Threshold graphs for each classifier. Clockwise from top left: logistic regression, neural network, random forest, gradient boost . . . . .	9
2	Reliability plots for each classifier. Clockwise from top left: logistic regression, neural network, random forest, gradient boost . . . . .	11
3	Cross-Validation Accuracy . . . . .	14
4	Cross-Validation Total Cost . . . . .	14
5	Error cost and price ratios for both misclassification types (left); Ratio of total True Negatives and False Negatives per price bin (right). . . . .	16

## Abbreviations

<b>ANN</b>	Artificial Neural Network
<b>EM</b>	Ensemble Model(s)
<b>LM</b>	Linear Model(s)

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Previous Literature</b>	<b>2</b>
2.1	Quantifying the Problem . . . . .	2
2.2	Model Infrastructure . . . . .	3
2.3	Analyzing Customer Behavior . . . . .	3
<b>3</b>	<b>Methodology</b>	<b>3</b>
3.1	Heterogeneous ensemble models . . . . .	3
3.2	Base classifier models . . . . .	4
<b>4</b>	<b>Data</b>	<b>5</b>
<b>5</b>	<b>Feature engineering</b>	<b>5</b>
5.1	Feature construction . . . . .	5
5.2	Feature selection . . . . .	7
<b>6</b>	<b>Model Building</b>	<b>8</b>
6.1	Performance Measurement . . . . .	8
6.2	Experimental Design . . . . .	9
6.2.1	Nested Resampling . . . . .	9
6.2.2	Model Calibration . . . . .	10
6.3	Empirical Results . . . . .	12
6.3.1	Ensembling . . . . .	15
6.3.2	Post-hoc Cost-Sensitive Prediction Adjustment . . . . .	15
<b>7</b>	<b>Results</b>	<b>16</b>
<b>8</b>	<b>Conclusion</b>	<b>16</b>
<b>9</b>	<b>References</b>	<b>17</b>

# 1 Introduction

A major obstacle for companies participating in the rapidly rising E-Commerce sector are excessive product returns. Up to 35 % of the total sales in high fashion apparel are affected by this phenomenon (Toktay, Laan, and Brito 2004).

Historically, rising product returns have been often regarded as a necessary evil that is a byproduct of equivalently rising sales (Petersen and Kumar 2009). Nowadays, having customer- and product specific data en masse in combination with easily available computational power and sophisticated machine learning algorithms, there is no more need for companies to remain blind towards unnecessarily high rates of product returns and the associated risk.

The possibility of free returns may be regarded as a result of fierce online competition, to enhance customer satisfaction and loyalty or to decrease risk on the buyer side and consequently pushing revenues. Another common argument in favor of the matter is to reduce uncertainty of a potential customer concerning product (Heilig, Hofer, Lessmann, and Voß 2016), which is generally higher in comparison to the haptic product experience in a brick-and-mortar store (Haptics Paper).

On the other hand abnormally high product return rates are reported to have substantial impact on a company's profitability. Therefore it would be desirable to create a tradeoff between the mentioned advantages on the customer side and the losses on the corporate side. In order to do so, the drivers of product returns must be laid out in advance. These are often reported to be uncertainty over a product, recurrently expressed as a missfit, fraudulent behavior in terms of using the delivered item for a certain occasion and returning it afterwards as well as damage inflicted by the customer or defects (Potdar 2009).

Several measures intended to encounter such behavior have been proposed by Urbanke, Kranz, and Kolbe (2015). Suggested counter-measures include increased delivery time, moral suasion or transaction denial. Proposing limited payment options is in line with Pur et al., who find that Vorkasse, Sofortueberweisung and Lastschrift present a higher restriction to product returns and therefore decrease the likelihood of such (todo: accurately translate these terms).

Some of these options are already in practice for multiple large online retailers (<http://www.retailwire.com/discussion/should-chronic-returners-be-banned/>, <https://www.nbcchicago.com/news/local/Amazon-Bans-Woman-For-Life-Over-Too-Many-Returns-388199532.html>)

However, the measures to be taken are out of scope of this paper. Instead a underlying prediction system will be developped to analyze the drivers of product returns and predict them accurately. Such a forecast may supply companies with information on the reverse product flow and increase their inventory control (Brito and Laan 2009).

## 2 Previous Literature

### 2.1 Quantifying the Problem

[http://www.ibi.de/files/Studie\\_Retourenmanagement-im-Online-Handel\\_-\\_Das-Beste-daraus-machen\\_Summary.pdf](http://www.ibi.de/files/Studie_Retourenmanagement-im-Online-Handel_-_Das-Beste-daraus-machen_Summary.pdf)

Pur et al. report a lack of consciousness of many companies towards the risk and profit options caused by product returns. Hence almost 40% of the E-Retailers are not aware of the according cost. They identify the main reason for product returns to be lack of fit. A third of the surveyed companies expect their profit to rise by 5% in case of 10% less returns.

## 2.2 Model Infrastructure

Heilig, Hofer, Lessmann, and Voß (2016) and other lessmann paper enrich the literature with applied ensemble models in churn and customer return prediction tasks. They show that the overall performance is generally superior to each of the constituent's contribution. The accuracy increases with higher diversity among the underlying models. Additional insight in the construction of data-driven prediction systems are provided, as linear data growth may cause exponentially increased computation time.

Urbanke, Kranz, and Kolbe (2015) present a new framework to large-scale feature extraction. They mitigate common problems caused by sparse data matrices using dimensionality reduction in terms of Mahalanobis Feature Extraction. They report an outperformance compared to standard machine learning techniques.

## 2.3 Analyzing Customer Behavior

<http://www.prism.gatech.edu/~bt71/articles/forecasting.pdf> First - Toktay [https://uta-ir.tdl.org/uta-ir/bitstream/handle/10106/2001/Potdar\\_uta\\_2502D\\_10410.pdf?sequence=1](https://uta-ir.tdl.org/uta-ir/bitstream/handle/10106/2001/Potdar_uta_2502D_10410.pdf?sequence=1) Second - Potdar

Toktay concludes that, despite the economic impact, literature on predicting product returns is generally sparse. Existing models are criticized to disregard relevant influential variables as product price, rebates, environmental consciousness and return price. He emphasizes the possibilities of accurate product return predictions on supply chain management and inventory control.

The latter is picked up by Potdar (2009), who classifies product returns in the Consumer Electronics Industry according to the reason for which they have been returned. Especially Data Envelopment Analysis and Linear Regression are used for reason-specific analysis.

Badea (2014) uses ANN to predict consumer behavior in Romania. Using a publically available dataset on household as well as individual level, it is investigated whether the individuals would save money using bank deposits under certain conditions. The resulting predictions of the ANN are found to substantially increase accuracy compared to the ones of classical discriminant analysis.

Todo:

Lessmann Paper

# 3 Methodology

## 3.1 Heterogeneous ensemble models

The idea behind ensemble models (EM) is to combine multiple learning algorithms that taken together improve prediction accuracy over the performance of the individual models. The success of EM in predictive analytics



is due to their ability of reducing both bias and variance by combining multiple model forecasts (Dietterich 2002). EM can be divided into homogeneous and heterogeneous approaches. Homogeneous EM use a single base-learning algorithm (e.g. decision tree) on distinct training sets. In contrast, heterogeneous EM train and combine learning algorithms of different types. In the context of classification problems, heterogeneous EM are also referred to as multi-classifier systems (MCS) and have gained increasing popularity (Ranawana and Palade 2006). Two key design features for the success of MCS is the diversity among the selected classifiers as well as the good individual performances (strength) of the baseline classifiers (Ali, Sremath Tirumala, and Sarrafzadeh 2015). However, it has been shown that neither diversity nor individual performance of baseline models can directly predict the ensemble success (Ranawana and Palade 2006).

MCS have been shown to be useful in predicting customer e-commerce behavior, outperforming single classifiers in prediction accuracy (Tsoumakas, Partalas, and Vlahavas (2008), Kim, Kim, and Lee (2003)). Specifically, Heilig, Hofer, Lessmann, and Voß (2016) demonstrate the suitability of an ensemble selection approach (ES) for predicting product returns in online retailing. This three-step ensemble technique builds upon the creation of a library of heterogeneous candidate base models. Each classifier method can be used to build up multiple models with different meta-parameter settings. The ensemble model is created by combining base model from the library through an extensive search strategy. Though this approach has been demonstrated to be very effective for accurate predictions, it nevertheless requires extensive computational time. We therefore adopt a standard two-step ensemble approach. First, we develop and train four base classifiers models. Second, we combine these individual models with a simple non-linear combination method, namely unweighted majority voting. We opt for this majority voting technique since because it is computationally inexpensive and is likely to improve model accuracy in cases when individual baseline models are well trained (Kuncheva 2004). In case of a voting tie, the base model with the highest single accuracy will act as a tiebreaker.

### 3.2 Base classifier models

The ensemble model consists of four separately trained constituents as summarized below. They have been selected due to their outstanding performance as classification models. Since the relationship between product returns and the independent variables is suspected to be of non-linear nature, the majority of the base models captures effects of such nature. Nevertheless, as information heterogeneity substantially influences the performance of an ensemble model, a logistic regression is used to capture linear relationships among the variables. An ANN is a statistical model inspired by the functionality of the human brain. It consists of an input layer, possibly multiple hidden layers and an output layer, in which an activation function sends a signal dependent upon the processed input. According to the model structure, ANN can produce varying results (Qin et al. 2013). Pacelli and Azzolini (2011) figure ANN successful in unveiling complex relationships, especially in combination with linear models, but emphasize the need to train them extensively for the risk of remaining in local minima during minimization of the error function. Following Qin et al. (2013), the inclusion of a random forest model enriches the information drawn from ANN and logistic regression as it does not rely on estimated predictors and is not prone to overfitting. With respect to the large amount of data and the computational intensity of ANN, random forests are an efficient model improvement. TODO: Gradient Boost Stuff

	Models	Relationship
1	Logistic Regression	linear
2	Neural Network	nonlinear
3	Random Forest	nonlinear
4	Gradient Boost	nonlinear

Table 1: Baseline Models

## 4 Data

The two data sets available to us contain a total of 150,000 order records from an online apparel retailer from a yearlong selling period. For 50,000 of these records, it is unknown whether an ordered item has been sent back by the customer or not. This second data set is the target of our binary predictions on customer returning behavior (return/not return). Both data sets include a total of 13 continuous/discrete and categorical variables. These covariates give information on customer demographics (e.g. user state, date of birth, title), order details (e.g. order date, delivery date), and item characteristics (e.g. item size, price or color). The majority of customers are middle-aged female shoppers, with a mean order volume of 70 Euro over the entire record period and a mean product return rate of 48%.

To prepare the data sets for our analysis, we apply a set of standard pre-processing actions. Following the careful inspection of each variable, we remove all implausible values (e.g. extreme outliers). Approximately 20% of all records have missing values in the variables *delivery date* and/or *date of birth*. For better comprehensibility, we transform these covariates to *delivery time* and *age* respectively. Before imputing missing values, we flag them using dummy variables to extract their predictive power (Lessmann 2016, 18). Since *age* seems to be missing completely at random according to our data inspection, imputing it using mean substitution gives us an unbiased estimates (Schafer and Graham 2002). Missing values in delivery time, caused by missing delivery dates, are clearly not missing not at random as they have a zero mean return rate and therefore are a perfect predictor. Possible reasons for these missing values are manifold. Without knowing the process generating these missings, we cannot find unbiased substitutes for them (Schafer and Graham 2002, 171). We adopt three single substitution methods, namely case dropping, mean and median imputation, and chose the latter one based on model performance. We standardize the continuous variables only after the feature creation step to maintain their interpretability. Likewise, we do not directly drop zero (almost) zero variance predictors (e.g. date of birth) since we use them for feature extraction.

## 5 Feature engineering

### 5.1 Feature construction

Comprehensive feature engineering, i.e. creating and selecting useful and informative features, is the key to successful e-commerce prediction models (Liu et al. 2016). Based on the data structure available we separate our indicators into three e-commerce-related categories, namely product, customer and basket related features. Based on the data structure available we separate our indicators into three e-commerce-related categories, namely product, customer and basket related features as can be seen in table 2. Features in the

product category are all item-specific indicators (e.g. item price). Likewise, customer-related features contain variables with relational or demographic information on each customer (e.g. customer age). We define the basket-related features as those that are common within one order basket (e.g. number of items within a basket). We consider all items belonging to one basket that were ordered by the same user on the same day.

Table 2: Constructed features ordered by category (excerpt)

Feature category	Feature	Type	Description
Product	Item price	Numeric	Price at which item is purchased
	Item category	Factor	Category (eg. clothing, shoes) of item
	Item WOE	Numeric	WOE based on item id
Customer	Customer Age	Numeric	Age in years at purchase
	Account Age	Numeric	Days since registration at purchase
	User’s income	Numeric	Proxy for costumers’s income
Order/Basket	Large basket	Factor	No. of items in same basket exceeds one
	Discount pc.	Numeric	Discount on item price in %
	Same category	Numeric	No. of items belonging to the same item category within same basket

In order to extract as much information as possible from the 13 original variables, we apply a variety of different extraction and projections methods, including unsupervised and supervised methods. Our extraction methods can be broadly summarized in four groups: discretization, projection, transformation/combination of variables and data augmentation by using external information. Based on these methods we construct an enriched data set with a total of 45 features. A small excerpt of these features can be seen in table 2. We apply discretization to several continuous variables, as this can be useful for capturing non-linear effects, constructing a more comprehensible representation of variables or improving predictive accuracy, especially in the context of tree-based decision algorithms (Liu et al. 2002). For example, we construct price item bins using equal frequency discretization as we suspect non-linearity regarding the target variable. This approach is superior to equal width discretization for our purpose since it is outlier-resistant and gives us a higher interval resolution for item prices with a high density (Cichosz 2014, 531). A special challenge of this data set, which applies to many predictive modeling settings on customer behavior, is the presence of high-cardinality attributes (e.g. user id). We address this by replacing each category with its Weight of evidence (WOE), a numerical projection method for categorical variables. The WOE is a measure of predictive power of a category with respect to the target variable, which has gained popularity in credit scoring (Lessmann 2016, 19–20). For a given customer, it shows a customers tendency in returning behavior, a valuable predictor variable. Since WOE has been shown to work well across the different types of our baseline models, requires low computational effort and reduces dimensionality(Lessmann 2016, 32–33), we apply this projection methods to all categorical variables. For categories where WOE is not computable due to class sparseness, we add artificial data points as suggested by Kohavi and John (1997).

Another useful approach we take is the transformation and combination of variables based on domain knowledge about consumer’s shopping and returns behavior (Arnold and Reynolds 2003). For example, we expect that

discounted items will have a lower return rate due to lower item quality expectation as proposed by Petersen and Kumar (2009). Thus, we construct several discount variables (e.g. absolute discount, percentage discount) by taking the maximum item price for each item as proxy for the regular item price and considering all deviations as a discount. Furthermore, a common defect of online shopping is that customers might be unsure about which size or color they should order based on the item description and thus place multiple orders of the same item (Foscht et al. 2013). We therefore construct informative variables on the order basket of consumers, e.g. the basket size in terms of items or the number of similar items with distinct sizes within one basket. Furthermore, we augment and enrich the data sets by including external sources to construct new features. For example, we use statistics on the mean income for different age groups and different regions within Germany. By combining these indicators with each customer’s demographical data (age and user state) we are able to construct a proxy for each customer’s income. Moreover, we tackle the lack of information on item categories (e.g. pants, shoes, clothing, accessories). Based on size tables for different apparel categories we gathered we are able to categorize over 90% of items by finding sizes unique to each category. As expected, the mean return rate of shoes (52%) is almost twice as high as mean returns in accessories (28%).

## 5.2 Feature selection

Finding an optimal feature subset is important for various reasons: First, learning algorithms like neural networks or tree-based models have been shown to degrade in performance when confronted with datasets containing redundant or irrelevant variables (Zdravevski et al. 2014). Second, feature selection reduces data dimensionality, reducing the needed computational power for model building process and improving model comprehensibility. We adopt a hybrid feature selection strategy by combining filters and wrappers. As a first preprocessing step as proposed by Zdravevski et al. (2014), we remove very poor variables based on model agnostic information criteria. For numerical variables we compute the F-Score as proposed by Chen and Lin (2006), whereas the average information value (obtained from WOE) is computed for categorical covariates. The second step in our feature selection is a wrapper approach as described in Kohavi and John (1997). This Greedy selection strategy is advantageous since it has been shown to be robust against overfitting and is designed to optimize model performance (Guyon and Elisseeff 2003, 1167). We use a sequential floating backward selection (SFBS) algorithm as proposed by Pudil, Novovičová, and Kittler (1994). SFBS is able to capture interaction effects between variables better than non-floating approaches and has been shown to provide good results in medium-scale problems like this (Kudo and Sklansky 2000). Since the single baseline models from our ensemble model might perform best on distinct feature subsets, the optimal approach would be to conduct an individual wrapper for each model. Due to computational constraints we are nevertheless limited to conducting a single wrapper for all models. As feature subsetting is especially important for tree-based models and neural networks, we design a wrapper based on a simple random forest. The experimental setup is similar to our inner resampling approach during baseline model training (section 6.2.1) The SFBS leaves us with a final subset of 17 features. nested resampling

## 6 Model Building

### 6.1 Performance Measurement

To evaluate predictive accuracy, we first consider two performance measures - classification accuracy and logarithmic loss. Classification accuracy is the number of correctly classified observations divided by the total number of observations. We use this initial measure due to its ease of interpretability. However, this measure does not differentiate between types of error (which is important due to asymmetric costs) nor does it penalize overconfidence in classification. Therefore we look additionally at the logarithmic loss, a form of cross-entropy that is used when the desired outcome is the probability that an example is positive. The formula for logarithmic loss is shown below, where  $N$  is the total number of observations,  $y_i$  is the true outcome, and  $p_i$  is our predicted probability outcome. This measure is designed to penalize overly confident but misclassified observations and is often preferred for binary classification problems like ours (Caruana and Niculescu-Mizil 2004).

$$logloss = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

Finally, we consider predictive accuracy within the framework of our specific prediction task, which is to minimize the cost associated with incorrect classifications. Specifically, our problem deals with example-dependent misclassification costs because the penalty for false classification is associated with the item's corresponding cost due to restocking fees or return fees. Namely, in the event of false positive classification (we incorrectly predict the customer will return), there will be an opportunity cost of a lost potential purchase of  $-0.5 * itemprice$ . In the event of a false negative classification (we incorrectly predict the customer will not return), we incur an actual financial penalty of  $-0.5 * 5 * (3 + 0.1 * itemprice)$ . For our purposes, we consider both actual and opportunity costs as impacting our cost calculations.

To solve this problem, we use a thresholding approach. The thresholding approach selects the probability cutoff point that minimizes the total misclassification cost from the training data set and uses it to predict the class label of the test instances. Test instances that have a predicted probability below or equal to this threshold are classified as a negative instance, whereas test instances with a probability higher than the threshold are classified as a positive (i.e. return) instance. Despite its simplicity, thresholding often empirically outperforms other cost-sensitive learning methods (Sheng and Ling 2006).

Our specific approach mirrors a similar approach as suggested by Fawcett (2006), in which he generates a ROC curve that is sensitive to instance-varying costs. In our approach, we loop through a sequence of possible threshold values between 0 and 1, plot the corresponding accuracy and cost, then select the threshold that minimizes the overall cost. Since costs are shown as negative, we seek to maximize the value. We plot these graphs showing the accuracy and costs side by side given different threshold points for each of the four classifiers in [FIGURE XXX]. Notice how the threshold that optimizes costs is different from the one that maximizes accuracy.

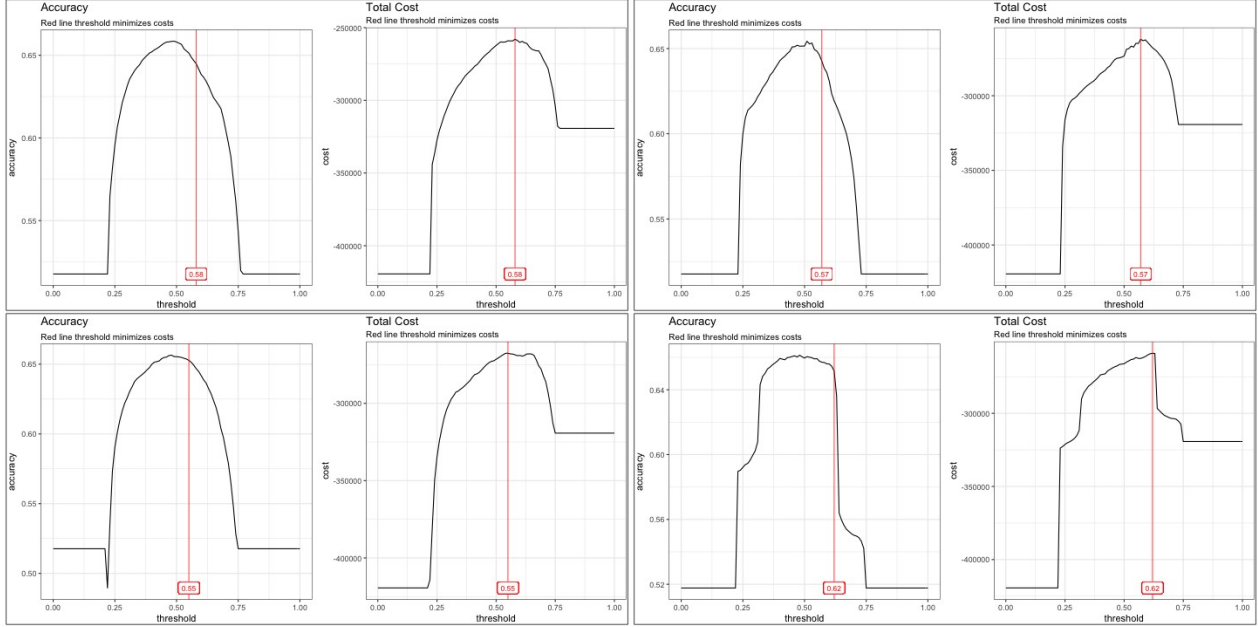


Figure 1: Threshold graphs for each classifier. Clockwise from top left: logistic regression, neural network, random forest, gradient boost

## 6.2 Experimental Design

### 6.2.1 Nested Resampling

In order to obtain reliable out-of-sample performance estimates for each classifier and the overall ensemble model, we undertake a nested resampling approach. This approach employs two cross-validation loops. We use the inner loop to tune the hyperparameters, while we use the outer loop to estimate how the model will generalize to a new, unknown data set by averaging the test results (Hastie, Tibshirani, and Friedman 2001, 241).

In the outer loop, we use 4-fold cross-validation. In this approach, we partition the initial test set containing 100,000 observations into four equally sized groups. For each iteration of the loop, we set aside the  $i$ -th partition for testing, while we use the remaining 4 partitions to train the individual classifiers (logistic regression, neural network, gradient boosting, random forest).

In the inner resampling loop, we train each classifier using a fixed holdout instance to find the optimal hyperparameters on each of the individual classifiers. We use a fixed holdout set to enable us to simply standardize numeric variables and create our Weight of Evidence features (by calculating off the training set and applying those values to the holdout set) within the inner loop. This preprocessing step is necessary for all variable transformations that are dependent on the data to prevent overfitting the model.

The inner resampling loop assesses the quality of the model given the chosen hyperparameters. With the consideration that a grid search across the entire hyperparameter search space would be too costly, we choose to instead do a random search across the search space. The random search method will randomly choose from our specified values, with a maximum of 20 iterations. Additionally, we train each classifier to optimize for

AUC (area under the ROC curve), since such classifiers tend to produce better accuracy relative to those that only optimize for accuracy (Huang and Ling 2005). The chosen hyperparameters are then passed to the outer loop where a new model is trained with the corresponding hyperparameters. We then predict the results on the outer loop holdout set and aggregate results.

### 6.2.2 Model Calibration

Additionally, in the outer loop, we calibrate the probability estimates generated from each classifier. We use calibration for two reasons: Firstly, it ensures that the probability estimates better reflect their true posterior probabilities (Caruana, Niculescu-Mizil, and Munson 2006). For example, in a well-calibrated model, if the probability estimate for a given instance is 70%, that means we can assume that that event has a 70% chance of occurring. Secondly, calibration transforms predictions from all models to place on them equivalent scale. Combining calibrated models in ensembling not only provides improvement in probability measures like logarithmic loss, but failure to use calibrated models may hurt ensemble performance (Caruana and Niculescu-Mizil 2005b).

We use a method known as Platt scaling to obtain calibrated predictions. In Platt scaling, we pass the output probabilities through a sigmoid (which in our case is a logistic regression) (Platt 1999). We apply the specific method below as described in (Caruana and Niculescu-Mizil 2005a) in order to prevent overfitting and unwanted bias:

1. Split the data set into a training and a test set
2. Split the training data set further into a calibration set
3. Train the model on the training set
4. Predict the model on the calibration set and the test set
5. Use the predictions from calibration set to train a logistic regression model
6. Pass the predictions from the test set through the logistic regression model in step 5 to obtain the final predictions

We visualize the effect of calibration on each of our classifiers using reliability diagrams from the first fold of cross-validation. Reliability diagrams split the prediction space into 10 bins, then plot the mean prediction value for each bin against the true fraction of positive cases. The diagonal line indicates a perfectly calibrated model (Caruana and Niculescu-Mizil 2005b). Additionally, in each diagram, we plot the histogram of predicted values before and after calibration. The results of calibration for each classifier can be seen in [FIGURE xxxx].

From the reliability plots, we can see a few major effects of calibration. Firstly the results after calibration (the line in blue) shift away from a sigmoidal shape (the line in red) to line up closer to the diagonal line indicating a perfectly calibrated model. Secondly, calibration pushes the probabilities closer to the center and away from 0 and 1, to better align with the true fraction of positive cases and prevent overconfidence in classifications. One benefit that we can recognize from the reliability plots is that overall, each histogram post-calibration appear similar to one another because calibration reduces the differences between predicted probabilities for each model, which will provide usefulness in ensembling (Caruana and Niculescu-Mizil 2005b).

Overall, we see benefits in calibration when comparing the accuracy, log-loss, and cost metrics. In [FIGURE xxx], we compare the average across all validation runs for the aforementioned metrics. We see that calibration yields significant improvements in log-loss error for all models without negatively impacting accuracy and cost

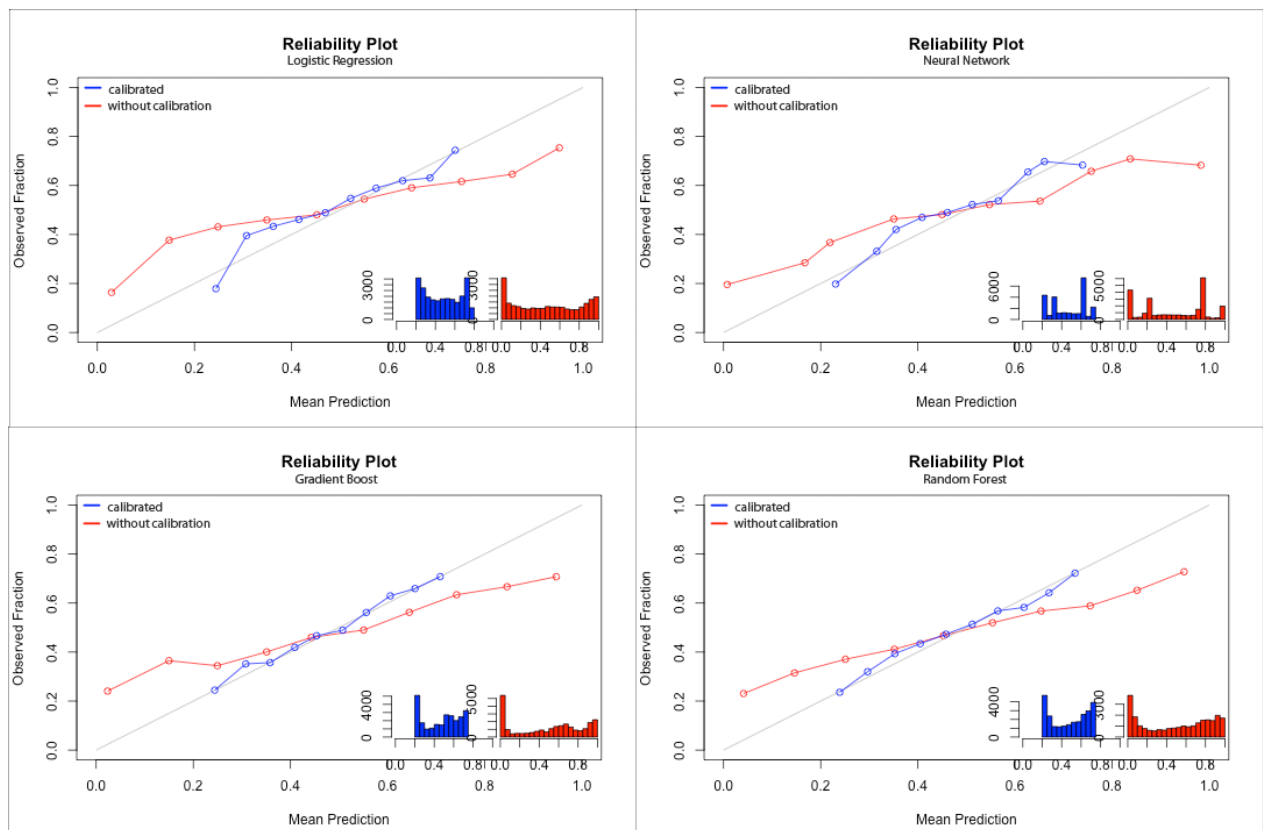


Figure 2: Reliability plots for each classifier. Clockwise from top left: logistic regression, neural network, random forest, gradient boost



measurements.

Table 3: Accuracy comparison

Classifier	Uncalibrated	Calibrated
Logistic Regression	0.657	0.657
Neural Network	0.662	0.661
Random Forest	0.656	0.656
Gradient Boost	0.653	0.654

Table 4: Log-loss error comparison

Classifier	Uncalibrated	Calibrated
Logistic Regression	0.709	0.618
Neural Network	0.774	0.621
Random Forest	0.774	0.625
Gradient Boost	0.817	0.628

Table 5: Total cost comparison

Classifier	Uncalibrated	Calibrated
Logistic Regression	-257376.4	-257644.0
Neural Network	-257108.2	-262476.1
Random Forest	-262464.2	-257532.9
Gradient Boost	-258767.0	-258816.6

### 6.3 Empirical Results

The output of the nested resampling results in a total of five candidate models and sets of predictions for each of the four classifiers. We use the calibrated results of these classifiers for the following:

1. Assess the stability of the hyperparameters
2. Estimate the prediction accuracy and total cost of the model
3. Get the average optimal decision threshold to apply to our final classifiers
4. Use the model with the lowest average cost as a tiebreaker in our ensemble

We can examine the stability of our hyperparameters in [TABLES x,y,z]. Since the hyperparameters do not vary wildly between iterations, we can make the assumption the inner loop optimization works as designed. The distribution of performance values (in terms of accuracy and total cost, respectively) across resampling iterations and for all classifiers is shown in [FIGURES x,y]. The dot in the center represents the average value, whereas the spread is indicated by 1 standard error above and below the mean. On average, the neural network consistently performs best both on prediction accuracy and total cost metrics. In comparison,

gradient boosting tends to perform the worst in terms of prediction accuracy but random forests perform the worst in terms of total cost minimization. Logistic regression, as expected, shows the smallest variation between iterations. However, although neural networks and logistic regression tends to vary little between iterations, gradient boosting and random forest approaches have far larger variations between iterations in terms of total cost minimization. The best performance of any one run came from a gradient boost iteration.

Table 6: Neural Network Hyperparameter Tuning

Iteration	size	decay
1	4	1e-03
2	5	1e-05
3	8	1e-02
4	6	1e-03

Table 7: Random Forest Hyperparameter Tuning

Iteration	ntree	mtry	nodesize
1	300	3	15
2	350	3	30
3	250	3	25
4	400	3	15

Table 8: Gradient Boost Hyperparameter Tuning

Iteration	gamma	eta	nrounds	lambda	max_depth
1	4	1e-03	100	0.3	3
2	5	1e-05	200	0.1	3
3	8	1e-02	100	0.4	3
4	6	1e-03	100	0	4

- Note from Claudia : Cite paper by Prof Lessman Coussement, Lessmann, and Verstraeten (2017): We explain good performance of log reg with existence of linear effects and our meticulous data preparation. As Coussement, Lessmann, and Verstraeten (2017) have demonstrated, logistic regression can compete with more complex algorithms ifexposed to well prepared data set

We note that nested resampling is a computationally intensive process. We limit the inner loop cross-validation process to 4 iterations for this reason. Additionally, we employ relatively small search spaces for hyperparameters (usually not exceeding 30 possible combinations) that are informed by theoretical considerations (citation???). In order to accelerate the overall model fitting process, we employ a parallelization wrapper around the tuning process.

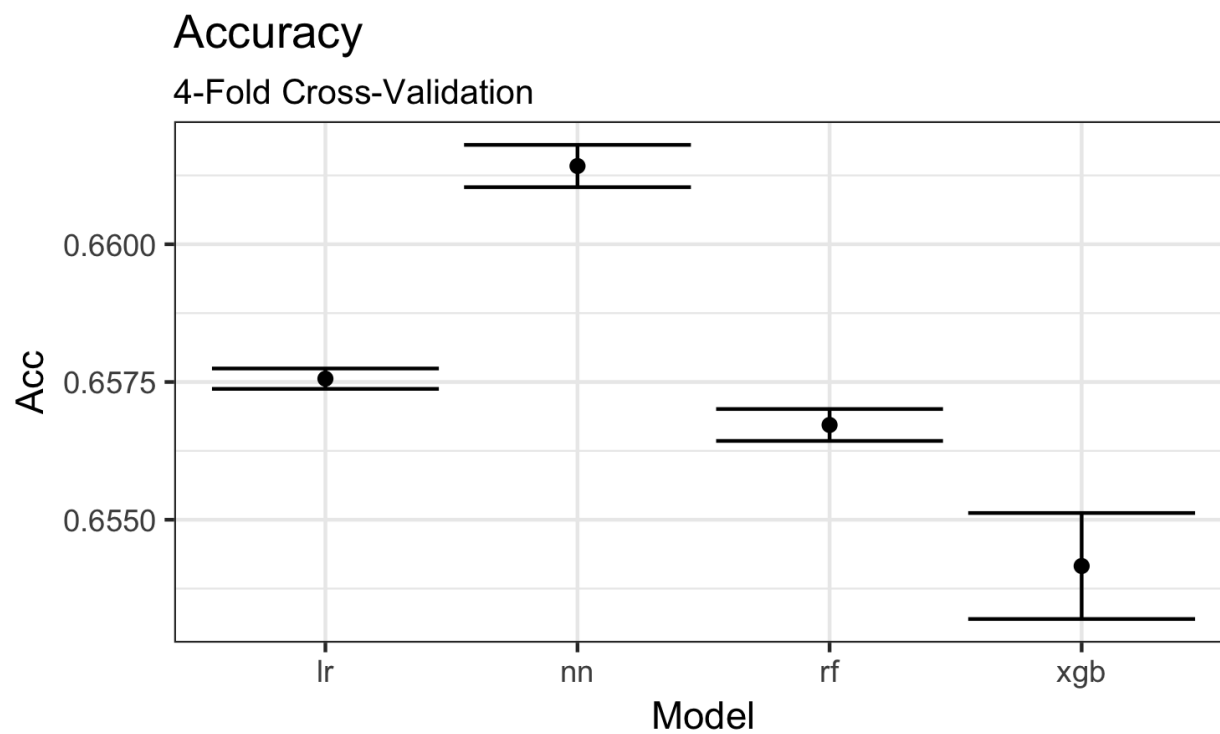


Figure 3: Cross-Validation Accuracy

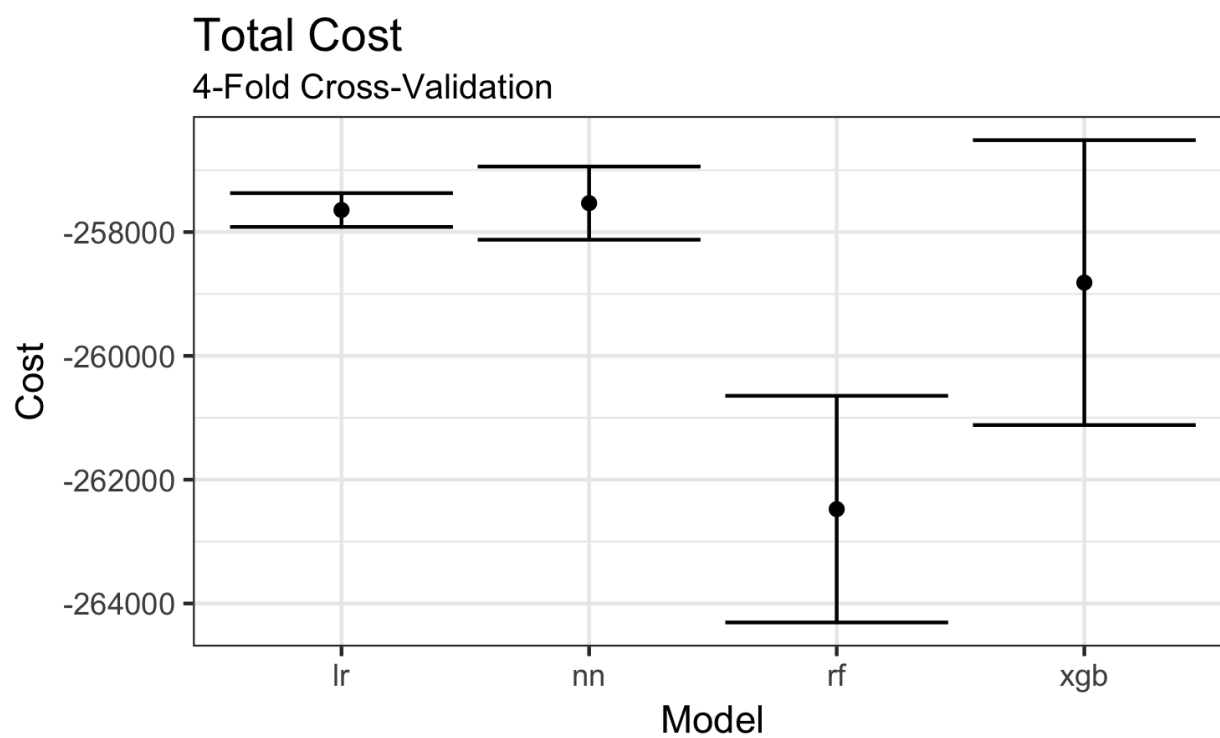


Figure 4: Cross-Validation Total Cost

### 6.3.1 Ensembling

Finally, we ensemble models from each of the  $k$  runs and compare the final combined result to the average of the individual candidate classifiers. Our ensemble approach uses majority voting as described in the Methodology section of the paper. Given that we are ensembling four models, we use the best model in terms of cost minimization as a tie breaker. In our case, our best performing model is the neural network.

For each of the 4 iterations of the cross-validation process, we use our ensembling method to assign final predictions for each instance. We then calculate the total cost of misclassification and compare that to the average cost for each individual classifier. The results can be seen in [TABLE XXXX].

Table 9: Cost Improvement from Ensembling

Iteration	Average Classifier	Ensembled	% Improvement
1	-261651.96	-256435.62	0.0199
2	-256735.30	-251296.48	0.0212
3	-258906.22	-255041.46	0.0149
4	-259175.97	-257744.25	0.0055

The results of ensembling show an average improvement of 1.5% compared to the average base classifier. Although this figure may seem small, it can result in significant cost savings for the retailer over the long run. Additionally, we note that in some cases the ensemble performs slightly worse than the best individual classifier. However, we believe this approach remains valid because of how high the variation is between individual runs, depending on the observations in the data set. The ensembling process helps reduce this variation by averaging the performance of each model.

### 6.3.2 Post-hoc Cost-Sensitive Prediction Adjustment

The final class memberships of our model predictions are the results of our cost-minimizing threshold strategy. Despite the suitability of threshold tuning in general, the outcome might not be optimal due to the presence of cost asymmetry. This holds especially true for the items with a low item price, where disparity between the distinct classification errors is the largest. This can be seen in figure 1 (left) where the error costs relative to the item price is displayed for both types of misclassification. We refer to those predictions as type error 1, if the predicted value ends up being a False Positive (FP), meaning that we erroneously predicted an item return to be returned by the customer. In contrast, the type 2 errors relate to the False Negatives which implies we predicted a non-return when the item has in fact been returned. For low item prices, the spread between both error cost ratios is very high because of the fixed return costs being relatively high to item price. We therefore suspected it may be cost-saving to manipulate our model predictions to reduce the costs of FN.

Specifically, we tested whether it is more costly to use our predicted model classes or to always predict customers to return the low price item and thus save on shipment costs (‘pure return strategy’). For a given item price, our pure return strategy saves costs if the savings in shipment costs outweigh the losses of items that would not have been returned by customers(TN). Thus, it must hold that  $0 < \frac{C_2}{C_1} - \frac{TN}{FN}$  where  $C_1$  and  $C_2$  refer to the misclassifications errors of type 1 and 2 at a given item price. Since some item price levels have

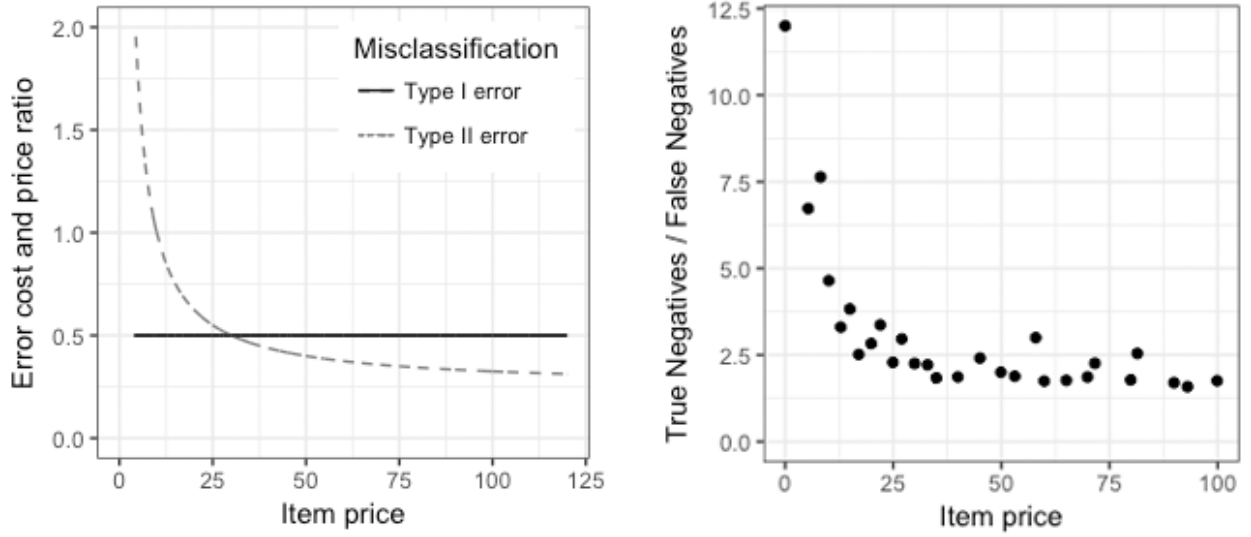


Figure 5: Error cost and price ratios for both misclassification types (left); Ratio of total True Negatives and False Negatives per price bin (right).

sparse class membership, we create equal frequency price bins and approximate C1 and C2 with mean errors costs. We test our strategy based on the baseline prediction of our logistic regression model on a stratified test set. In our sample, the pure return strategy only saves costs for a item price of zero. This is because the high spread between costs is outweighed by the high ratio between TN and FN. This can be seen in figure 1 (right), where this ratio is displayed for all price bins. As very few cheap items are returned by customers, the number of FN is small in low price bins, leading to a high ratio.

## 7 Results

## 8 Conclusion

## 9 References

- Ali, Shahid, Sreenivas Sremath Tirumala, and Abdolhossein Sarrafzadeh. 2015. “Ensemble Learning Methods for Decision Making: Status and Future Prospects.”
- Arnold, Mark J, and Kristy E Reynolds. 2003. “Hedonic Shopping Motivations.” *Journal of Retailing* 79 (2). Elsevier: 77–95.
- Badea, Laura Maria. 2014. “Predicting Consumer Behavior with Artificial Neural Networks.” *Procedia Economics and Finance* 15: 238–46. doi:[https://doi.org/10.1016/S2212-5671\(14\)00492-4](https://doi.org/10.1016/S2212-5671(14)00492-4).
- Brito, Marisa P. de, and Erwin A. van der Laan. 2009. “Inventory Control with Product Returns: The Impact of Imperfect Information.” *European Journal of Operational Research* 194 (1): 85–101. doi:<https://doi.org/10.1016/j.ejor.2007.11.063>.
- Caruana, Rich, and Alexandru Niculescu-Mizil. 2004. “Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria.” In, 69–78. ACM Press.
- . 2005a. “Obtaining Calibrated Probabilities from Boosting.” In *In: Proc. 21st Conference on Uncertainty in Artificial Intelligence (Uai '05)*, Auai Press. AUAI Press.
- . 2005b. “Predicting Good Probabilities with Supervised Learning.” In *Proceedings of the 22Nd International Conference on Machine Learning*, 625–32. ICML '05. New York, NY, USA: ACM.
- Caruana, Rich, Alexandru Niculescu-Mizil, and A. Munson. 2006. “Getting the Most Out of Ensemble Selection.” In *Sixth International Conference on Data Mining (Icdm'06)*, 828–33.
- Chen, Yi-Wei, and Chih-Jen Lin. 2006. “Combining Svms with Various Feature Selection Strategies.” In *Feature Extraction*, 315–24. Springer.
- Cichosz, Pawel. 2014. *Data Mining Algorithms: Explained Using R*. John Wiley & Sons.
- Coussement, Kristof, Stefan Lessmann, and Geert Verstraeten. 2017. “A Comparative Analysis of Data Preparation Algorithms for Customer Churn Prediction: A Case Study in the Telecommunication Industry.” *Decision Support Systems* 95. Elsevier: 27–36.
- Dietterich, Thomas G. 2002. “Ensemble Learning.” *The Handbook of Brain Theory and Neural Networks* 2. MIT Press: Cambridge, MA: 110–25.
- Fawcett, Tom. 2006. “ROC Graphs with Instance-Varying Costs.” *Pattern Recogn. Lett.* 27 (8). New York, NY, USA: Elsevier Science Inc.: 882–91.
- Foscht, Thomas, Karin Ernstreiter, Cesar Maloles III, Indrajit Sinha, and Bernhard Swoboda. 2013. “Retaining or Returning? Some Insights for a Better Understanding of Return Behaviour.” *International Journal of Retail & Distribution Management* 41 (2). Emerald Group Publishing Limited: 113–34.
- Guyon, Isabelle, and André Elisseeff. 2003. “An Introduction to Variable and Feature Selection.” *Journal of Machine Learning Research* 3 (Mar): 1157–82.
- Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.
- Heilig, Leonard, Julien Hofer, Stefan Lessmann, and Stefan Voß. 2016. “Data-Driven Product Returns

- Prediction: A Cloud-Based Ensemble Selection Approach.” *Research-in-Progress Papers* 34. Springer. [http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1033&context=ecis2016\\_rip](http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1033&context=ecis2016_rip).
- Heilig, Leonard, Julien Hofer, Stefan Lessmann, and Stefan Voß. 2016. “Data-Driven Product Returns Prediction: A Cloud-Based Ensemble Selection Approach.” In *ECIS*, Research-in.
- Huang, Jin, and Charles X. Ling. 2005. “Using Auc and Accuracy in Evaluating Learning Algorithms.” *IEEE Transactions on Knowledge and Data Engineering* 17: 299–310.
- Kim, Eunju, Wooju Kim, and Yillbyung Lee. 2003. “Combination of Multiple Classifiers for the Customer’s Purchase Behavior Prediction.” *Decision Support Systems* 34 (2). Elsevier: 167–75.
- Kohavi, Ron, and George H John. 1997. “Wrappers for Feature Subset Selection.” *Artificial Intelligence* 97 (1-2). Elsevier: 273–324.
- Kudo, Mineichi, and Jack Sklansky. 2000. “Comparison of Algorithms That Select Features for Pattern Classifiers.” *Pattern Recognition* 33 (1). Elsevier: 25–41.
- Kuncheva, Ludmila I. 2004. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons.
- Lessmann, Stefan. 2016. “Data Preprocessing in Database Marketing: Tasks, Techniques, and Why.” In *Advanced Database Marketing: Innovative Methodologies and Applications for Managing Customer Relationships*. Routledge.
- Liu, Guimei, Tam T Nguyen, Gang Zhao, Wei Zha, Jianbo Yang, Jianneng Cao, Min Wu, Peilin Zhao, and Wei Chen. 2016. “Repeat Buyer Prediction for E-Commerce.” In *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 155–64. ACM.
- Liu, Huan, Farhad Hussain, Chew Lim Tan, and Manoranjan Dash. 2002. “Discretization: An Enabling Technique.” *Data Mining and Knowledge Discovery* 6 (4). Springer: 393–423.
- Pacelli, V., and M. Azzolini. 2011. “An Aritifial Neural Network Approach for Credit Risk Management.” *Journal of Intelligent Learning Systems and Applications* 3. Scientific Research: 103–12. doi:10.4236/jilsa.2011.32012.
- Petersen, J Andrew, and V Kumar. 2009. “Are Product Returns a Necessary Evil? Antecedents and Consequences.” *Journal of Marketing* 73 (3). American Marketing Association: 35–51.
- Platt, John C. 1999. “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods.” In *ADVANCES in Large Margin Classifiers*, 61–74. MIT Press.
- Potdar, Amit. 2009. “Methodology to Forecast Product Returns for the Consumer Electronics Industry,” December. The Univeristy of Texas at Arlington. [https://uta-ir.tdl.org/uta-ir/bitstream/handle/10106/2001/Potdar\\_uta\\_2502D\\_10410.pdf?sequence=1](https://uta-ir.tdl.org/uta-ir/bitstream/handle/10106/2001/Potdar_uta_2502D_10410.pdf?sequence=1).
- Pudil, Pavel, Jana Novovičová, and Josef Kittler. 1994. “Floating Search Methods in Feature Selection.” *Pattern Recognition Letters* 15 (11). Elsevier: 1119–25.
- Qin, Qin, Qing-Guo Wang, Ji Li, and Shuzhi Sam Ge. 2013. “Linear and Nonlinear Trading Models with Gradient Boosted Random Forests and Application to Singapore Stock Market.” *Journal of Intelligent Learning Systems and Applications* 5. Scientific Research: 1–10. <http://dx.doi.org/10.4236/jilsa.2013.51001>.
- Ranawana, Romesh, and Vasile Palade. 2006. “Multi-Classifer Systems: Review and a Roadmap for

- Developers.” *International Journal of Hybrid Intelligent Systems* 3 (1). IOS Press: 35–61.
- Schafer, Joseph L, and John W Graham. 2002. “Missing Data: Our View of the State of the Art.” *Psychological Methods* 7 (2). American Psychological Association: 147.
- Sheng, Victor S., and Charles X. Ling. 2006. “Thresholding for Making Classifiers Cost-Sensitive.” In *AAAI*, 476–81. AAAI Press.
- Toktay, L.B., E.A. van der Laan, and M.P. de Brito. 2004. “Managing Product Returns: The Role of Forecasting,” December. Springer. [https://doi.org/10.1007/978-3-540-24803-3\\_3](https://doi.org/10.1007/978-3-540-24803-3_3).
- Tsoumakas, Grigorios, Ioannis Partalas, and Ioannis Vlahavas. 2008. “A Taxonomy and Short Review of Ensemble Selection.” In *Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications*.
- Urbanke, Patrick, Johann Kranz, and Lutz Kolbe. 2015. “Predicting Product Returns in E-Commerce: The Contribution of Mahalanobis Feature Extraction Completed Research Paper,” December.
- Zdravevski, Eftim, Petre Lameski, Andrea Kulakov, and Dejan Gjorgjevikj. 2014. “Feature Selection and Allocation to Diverse Subsets for Multi-Label Learning Problems with Large Datasets.” In *Computer Science and Information Systems (Fedcsis), 2014 Federated Conference on*, 387–94. IEEE.



## Declaration of Authorship

TEXT

15.01.2018