UNIVERSITEIT GENT

FACULTY OF ECONOMICS AND BUSINESS ADMINISTRATION

Advances and Applications in Ensemble Learning

Michel Ballings

2014

Advisor:
Prof. Dr. Dirk Van den Poel

Dissertation submitted to the Faculty of Economics and Business Administration, Ghent University, in fulfillment of the requirements for the degree of Doctor in Applied Economic Sciences

Typeset in LaTeX.

PhD Series - Ghent University, April 2014
Faculty of Economics and Business Administration
http://www.FEB.UGent.be

Divide et impera (divide and rule)
Julius Caesar

Many ensemble learning algorithms adopt a divide-and-conquer strategy in that each constituent learner only learns a smaller or simpler partition of the problem.

Doctoral jury

Acknowledgements

The contributions of many different people, in their different ways, have made this research possible. I would like to extend my appreciation especially to the following.

I wish to express my deepest gratitude towards my advisor, Prof. dr. Dirk Van den Poel. He offered me unequivocal support and high-quality advice throughout the entire process. He taught me invaluable skills and offered me countless opportunities enabling my development into an independent researcher. His guidance empowered me to gain my full potential. I would like to thank him for the first-class relationship during the years that I have been his assistant.

I am also greatly indebted to Prof. dr. Dries Benoit. His experience helped me to chart a course towards the researcher that I am today. He has consistently laid down markers enabling me to make informed decisions about software, productivity tools, and research.

I want to pay special tribute to my colleagues Willem Standaert and Dauwe Vercamer for the insightful discussions and fruitful collaborations. In addition I would like to thank them for the rich environment in which both productive interchange and sharing knowledge is valued.

Great gratitude goes to my wife Sanae Nechad, for the ample motivational support and help in finishing this project successfully. Without her this would not have been possible. I would like to thank my baby son, Alexander Ballings, for helping me keep things in perspective.

Special gratitude goes to my parents, Willy Ballings and Simonne Van de Velde, for instilling in me confidence and a drive for pursuing my PhD. I would like to thank my father for his expertise and excellent help with my personal computational resources that are partly used in this dissertation.

I am greatly indebted to my thesis students for the many interesting exchanges: Matthijs Meire, Matthias Lelie, Tessa De Backer, Ine Laleman, Matthias Bogaert, Charlotte De Baere, Steven Hoornaert, Vanessa Rys, Ruben Gryp, Nathalie Hespeels, Dieter D'Haenens, David De Winter, and Mathijs Depuydt.

I would like to express my gratitude to our head of department, Prof. dr. Patrick Van Kenhove, for fostering the stimulating academic environment in the department of Marketing. I would also like to thank the Psilogy team for providing startup funding for my PhD. Finally I acknowledge the department of Marketing for the excellence of their professors, the possibility to attend many international conferences, and the driven predoctoral researchers.

Ghent, April 2014, Michel Ballings

Table of Contents

List of Figures

List of Tables

Summary (Dutch)

Groepsleren, ook commissie-gebaseerd leren of het leren van systemen met meerdere classificatiemodellen, is ontstaan in de jaren 90 en is uitgegroeid tot een zeer belangrijk paradigma (Zhou, 2012). Groeps- leeralgoritmen worden gebruikt om problemen met evaluatie, optimalisatie, voorstelling en numerieke berekeningen op te lossen (Dietterich, 2000).

Ondanks de excellente voorspellende classificatieperformantie van groeps- leeralgoritmen worden deze vaak in diskrediet gebracht op het gebied van interpreteerbaarheid (De Bock and Van den Poel, 2012; Gareth et al., 2013). De mogelijkheid om het geleerde model te kunnen interpreteren wordt door meerdere auteurs aangeprezen als een van de basisvereisten voor een succesvol model (Qi et al., 2009; De Bock and Van den Poel, 2012). In klantenbeheersystemen, zijn interpreteerbare en intuïtieve modellen belangrijk opdat beslissingsnemers inzichten zouden kunnen verwerven in klantengedrag (Masand et al., 1999).

Deze dissertatie focust zowel op een aantal van de bovengenoemde sterktes als zwaktes van groepsleren. Studie 1 gebruikt groepsleren om berekeningsproblemen gerelateerd aan kernmatrices op te lossen aan de hand van een verdeel-en-heers techniek. Studie 2 vergelijkt de performantie van de techniek voorgesteld in studie 1 met andere algoritmen en toont eveneens aan dat de kritiek in verband met de interpreteerbaarheid van groepsleren onterecht is. Studie 3 focust op groepsleren om het voorstellingsprobleem op te lossen dat inherent verbonden is aan de industriestandaard ”Single Best”. De ”Single Best” techniek evalueert meerdere technieken en houdt enkel de beste techniek over voor het verdere proces. Studie 2 gebruikt deze industriestandaard. Studie 3 argumenteert dat deze technieken moeten worden gecombineerd aan de hand van groepsleren. In de volgende paragrafen vatten we de drie studies samen.

In studie 1 stellen we een groepsleermethode voor kernmachines voor. De leergegevens worden willekeurig opgesplitst in een aantal wederzijds exclusieve partities gedefinieerd door een rij- en kolomparameter. Elke partitie vormt een invoerruimte en wordt getransformeerd door een kernfunctie in een kernmatrix K. Vervolgens wordt elke K gebruikt als leergegevens om een binair ”Random Forest” basisclassificatiemodel te schatten. Dit resulteert in een aantal voorspellingen gelijk aan het aantal partities. Een gewogen gemiddelde combineert de voorspellingen in een finale voorspelling. Om de gewichten te optimaliseren wordt een Genetisch Algoritme gebruikt. Deze aanpak heeft het voordeel om tegelijkertijd (1) diversiteit, (2) nauwkeurigheid (3) en berekeningssnelheid te promoten. (1) Diversiteit is gecreëerd omdat de individuele Ks zijn gebaseerd op een subset van kermerken en observaties, (2) nauwkeurigheid

wordt nagestreefd door sterke ”Random Forest” modellen te gebruiken, en (3) berekeningssnelheid wordt bekomen omdat de berekening van elke K in parallel kan gebeuren. De performantie van het voorgestelde algoritme (”Kernel Factory”) wordt vergeleken met ”Random Forest” en ”Kernel- Induced Random Forest” aan de hand van vijf keer tweevoudige kruisvalidatie. De resultaten tonen aan dat ”Kernel Factory” significant beter presteert dan ”Kernel- Induced Random Forest”. Wanneer de juiste kernfunctie is gespecificeerd is ”Kernel Factory” ook significant beter dan ”Random Forest”. Een R-programma van het algoritme (*kernelFactory*) is beschikbaar gemaakt op CRAN.

Het doel van studie 2 is om (1) de haalbaarheid te toetsen van het voorspellen van de toename in gebruiksfrequentie van Facebook, (2) te evalueren welke algoritmes het beste presteren, en (3) inzicht te verwerven in welke variabelen belangrijk zijn. De performantie van ”Kernel Factory”, ”Random Forest”, ”Stochastic Adaptive Boosting”, Neurale Netwerken, ”Support Vector Machines”, en Logistische Regressie wordt vergeleken aan de hand van vijf keer tweevoudige kruisvalidatie. De resultaten tonen aan dat het haalbaar is om modellen te schatten met hoge voorspellende waarde. Het beste algoritme was ”Stochastic Adaptive Boosting” met een kruisgevalideerde AUC van 0.66 en een accuraatheid van 0.74. De belangrijkste voorspellers zijn onder andere de afwijking van reguliere gebruikspatronen, frequenties van specifieke gedragingen en groepslidmaatschappen, gemiddelde fotoalbum privacy instellingen, en de recentheid van geplaatste commentaar. Facebook en andere sociale netwerken kunnen voorspellingen van de toename in gebruiksfrequentie gebruiken om hun diensten aan te passen aan de gebruikers. Voorbeelden zijn het aanpassen van het tempo van advertenties en vriendschapsverzoeken, of zelfs het aanpassen van de persoonlijk nieuwsstroom. De grootste contributie van deze studie is dat het de eerste is om de toename in gebruiksfrequentie te voorspellen in een sociaal netwerk.

Het doel van studie 3 is om de toegevoegde waarde te toetsen van algoritme- geïnduceerde diversiteit over gegevens- geïnduceerde diversiteit in groepsleren. We ontwikkelen een Hybride Groepsleer- Algoritme bestaande uit zes sub- groepsleer algoritmen: ”bagged” Logistische Regressie, ”Random Forest”, ”Kernel Factory”, ”bagged Support Vector Machines”, ”Stochastic Adaptive Boosting”, en ”bagged” Neurale Netwerken. We testen het algoritme op elf gegevenssets aan de hand van vijf keer tweevoudige kruisvalidatie. Het Hybride Groepsleer-Algoritme is consistent en significant beter dan het ”Single Best” sub-groepsleer algoritme op alle gegevenssets wanneer de performantiemethode of het ”Self Organizing Migrating Algorithm” wordt gebruikt voor het schatten van de gewichten in de fusiemethode. Analyses tonen eveneens aan dat het Hybride Groepsleer- Algoritme grotere verbeteringen vertoont als de classificatietaak moeilijker wordt. Voor zover wij weten, is dit de eerste studie die de toegevoegde waarde van algoritme- geïnduceerde diversiteit toetst over gegevens- geïnduceerde diversiteit in groepsleren. Om onze resultaten gemakkelijk repliceerbaar te maken, hebben we R-programmatuur (genaamd *hybridEnsemble*) afgeleverd aan CRAN.

Referenties

De Bock, K. W., Van den Poel, D., Jun. 2012. Reconciling performance and interpretability in customer churn prediction using ensemble learning based on generalized additive models. Expert Systems with Applications 39 (8), 6816–6826.

Dietterich, T. G., 2000. Ensemble methods in machine learning. In: Kittler, J., Roli, F. (Eds.), Multiple Classifier Systems. Vol. 1857. pp. 1–15.

Gareth, J., Witten, D., Hastie, T., Tibshirani, R., 2013. An Introduction to Statistical Learning with application in R. Springer Texts in Statistics. Springer.

Masand, B., Datta, P., Mani, D. R., Li, B., Jun. 1999. CHAMP: a prototype for automated cellular churn prediction. Data Mining and Knowledge Discovery 3 (2), 219–225.

Qi, J., Zhang, L., Liu, Y., Li, L., Zhou, Y., Shen, Y., Liang, L., Li, H., Apr. 2009. ADTreesLogit model for customer churn prediction. Annals of Operations Research 168 (1), 247–265.

Zhou, Z.-H., 2012. Ensemble Methods: Foundations and Algorithms. Machine Learning & Pattern Recognition Series. Chapman & Hall/CRC, Boca Raton FL.

1 Extended Abstract

Since this is a PhD by publication, as opposed to a PhD as monograph, the manuscript is a collection of research papers which are organized in chapters. Every chapter can be read independently from the others.

The first chapter is an extended abstract in that it discusses the background of ensemble learning, research objectives, main findings, contributions, practical implications, limitations, and directions for future research. This chapter is meant to link the research papers to the common theme and provide and overview of the entire dissertation. After reading it, readers should be able to place all the remaining chapters in this dissertation in context.

1.1 Background

1.1.1 Motivation for Ensemble Learning

Machine learning algorithms, also called learners, consist of combinations of three components: representation, evaluation and optimization (Domingos, 2012). Representation stands for the structure of the learner (e.g., networks, trees), evaluation represents the objective function (e.g., accuracy, likelihood, squared error, information gain), and optimization denotes the method to search in a given representation for the highest performance of the objective function (e.g., greedy search, branch-and-bound, gradient descent). A learner inputs a data set of instances (x_i,y_i), also called patterns, objects or observations, with ($i = 1, 2, 3, \ldots, N$), where $x_i = (x_{i1}, \ldots, x_{in}) \in \mathbb{R}^n$ is the feature vector of instance i, $y_i \in$ {0,1} is the class label of instance i, n is the dimensionality of the input space, and N is the number of instances. The goal of a learner is to output a classifier, also called hypothesis or classification model[1].

Many different classifiers, can be learned from making specific combinations of input data, representation languages, evaluation functions and optimization methods. The resulting set of all possible classifiers that might be learned is called the hypothesis space $\mathcal{H}$. Consider the hypothesis space denoted by the outer curve in the left panel of Figure 1.1 (adapted from Dietterich, 2000). A learner searches the space $\mathcal{H}$ to identify the best hypothesis, $h \in \mathcal{H}$. The inner curve depicts a set of accurate hypotheses. Since one only disposes of a limited amount of data, it is possible that the learner finds different hypotheses (h_1, h_2, h_3) of equal *overall* accuracy. If in some region of the input domain, some hypotheses perform better than others, averaging reduces the risk of selecting the wrong hypothesis in a specific region. The point labeled g is the average of those three hypotheses and f is the true hypothesis. Point g is closer to f than any of the constituent hypotheses meaning that the combination of those points reduces the risk of selecting the wrong classification model. In this case combining or ensembling solves an evaluation problem (called a statistical problem by Dietterich, 2000). It has to be noted that there is no guarantee that the ensemble improves upon the best constituent classifier (Fumera and Roli, 2005). However, combining models does reduce the probability of selecting a poor model.

The middle panel in Figure 1.1 refers to how ensembling may solve optimization problems (called computational problems by Dietterich, 2000). Learners employ optimization methods for local searches (e.g., trees use greedy search) that may get stuck in local optima. When there are sufficient data and the evaluation problem does not exist, an algorithm may still struggle to find the best hypothesis. By averaging different hypotheses, obtained by starting local searches from different points, the true hypothesis f may be better approximated than any of the individual hypotheses.

The third problem that ensemble methods can alleviate is representational in nature (Dietterich, 2000). When $f \notin \mathcal{H}$, averaging can expand the space of representable functions as depicted in the right panel of Figure 1.1. For example, a linear learner cannot learn non-linear boundaries, while a combination of linear learners can learn any boundary. In essence, an ensemble effectively follows a divide and conquer approach in that each individual only learns a smaller or simpler partition of the problem. Note that very flexible learners such as neural networks and trees can represent all possible classifiers (i.e., f always falls in $\mathcal{H}$). However, this capability only holds asymptotically. In reality only a finite set of hypotheses can be generated given a finite input data set (Dietterich, 2000).

Computational problems such as data size limitations and execution time constraints, not depicted in Figure 1.1, can also be alleviated by ensemble methods. Instead of learning one model on the whole data set, multiple smaller models can be learned on different smaller subsets, as such reducing computer hardware requirements. Processing speed can also be increased since most ensembles are inherently parallel.

[1]This dissertation focuses on binary classification.

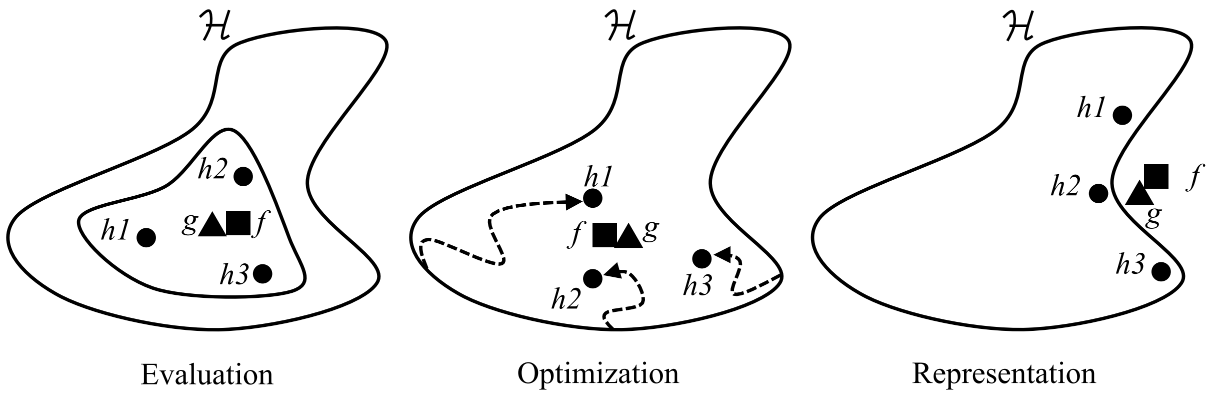


Figure 1.1: Reasons why an ensemble may outperform a single classifier

An algorithm that suffers from the evaluation issue or the optimization issue is said to have high variance. A algorithm suffering from the representational issue is said to have high bias. Hence, ensembling may reduce both variance and bias (Zhou, 2012, p67).

The next section covers the design considerations of ensemble learners. The two main topics are ensemble member generation and member combination.

1.1.2 Designing Ensemble Learners

Ensemble learning, also called committee-based learning or learning multiple classifier systems, has become a major paradigm since the 1990s and has roots in three distinct communities (Zhou, 2012). The pattern recognition community has mainly worked on combining strong classifiers. Research in this community has focused on designing powerful combination rules. In contrast, the machine learning community has mostly studied ensembles of weak learners, engendering methods such as Bagging (Breiman, 1996a) and Boosting (Schapire, 1990; Freund, 1995). Finally, the neural networks community has worked on mixtures of experts (ME). A ME is different from the other two approaches in that it employs a divide-and-conquer strategy while the other do not (or to a lesser degree). In ensembles of weak classifiers and ensembles of strong classifiers, the base learners, also called individual learners or team members, are trained on the same problem and similar data and are generally highly correlated (Zhou, 2012). In those cases the focus is on constructing diverse base classifiers. In ME, base classifiers are generated for different subtasks yielding inherently diverse individuals. The key problem is to find a natural divisions of the task, make the experts local, and derive a global solution from the individuals. An example of ME is when a pulmonologist has to combine the information in a PET scan and subject demographics. One classifier could be trained to learn the PET scan image and another one to learn the data set with subject demographics stored in more traditional features. Decisions of both models can then be combined by a combination rule (Parikh and Polikar, 2007).

All ensembles are created in two phases: base classifier generation and base classifier combination. To obtain a high performing ensemble the main focus in the generation step should be on creating accurate and diverse team members (Sharkey and Sharkey, 1997). On the one hand, combining the same classifiers does not yield better performance and only increases the system's complexity. On the other hand, different but much worse performing models are unlikely to increase ensemble performance neither (Ruta and Gabrys, 2005).

Diversity can be generated using either a data- strategy or an algorithm- strategy. The main focus of the former strategy is perturbing or partitioning the input data. For example, Bagging (Breiman, 1996a), short for bootstrap aggregating, creates diversity by generating multiple bootstrap samples. Another example is adaptive resampling, with Boosting (Freund, 1995) as its prime representative, that resamples data biased towards misclassified observations. The disadvantage of the latter is that it is iterative in nature whereas the other methods are parallel. In contrast to creating partitions on the instance level, the Random Subspace method (Ho, 1998), or attribute bagging (Bryll et al., 2003), focuses on the features. Random Forest (Breiman, 2001) combines both an instance- and feature- strategy. The algorithm- strategy mainly consists in using different learning algorithms in the ensemble (e.g., Tsoumakas et al., 2005)(resulting in a heterogeneous ensemble) or manipulating parameter settings for a given learner (e.g., Windeatt, 2005; Ueda, 2000). Both data-and algorithm-strategies result in a set of L hypotheses $(h_1, h_2, \ldots, h_L)$ that are aimed to be diverse and accurate.

In the second phase, base classifier combination, the first step is to choose the fusion rule. Depending on the output that is received from the base classifiers, different types of fusion rules exist. In case of binary classification, two levels of outputs can be distinguished: (1) class label outputs, and (2) scores, also called measurement level. The latter can be transformed into confidences, also called *a posteriori* probabilities, by applying a calibration function. For the former output type, majority vote is the most popular combination rule while the simple mean is the most popular for the latter. An overview of these and other methods can be found in Kuncheva (2004). Measurement level output usually is preferred as it contains the largest amount of information (Xu et al., 1992; Al-Ani and Deriche, 2002) and is able to reduce the total generalization error (Bauer and Kohavi, 1999). The goal of fusion functions is to integrate the different classifiers' output into one generalized output.

Measurement level output can be combined as follows. Each hypothesis (i.e., base classifier) $h_{(l,1)}(x)$ outputs the chance that x belongs to class 1. When using *a posteriori* probabilities, the fusion function $\mathcal{F}$ then gives the combined output $g_1(x)$ as the confidence that object x_i has class $y_i = 1$.

$$g_1(x) = \mathcal{F}(h_{(1,1)}(x), \ldots, h_{(L,1)}(x)) \tag{1.1}$$

Here, $\mathcal{F}$ can be based on (1) fixed rules or (2) trained weights (Roli et al., 2002). While fixed rules have a very small time complexity and provide simplicity, their result is expected to be worse than that of the trained ones. Depending on the size of the data and the number of classifiers available, one should carefully decide which one to take (Duin, 2002). For linear combiners, one can speak of a fixed rule if all the weights are equal (e.g., simple average). One can speak of a trained rule if the weights are different (e.g., weighted averaging) (Roli et al., 2002). A weight $w_{l,1}$ is given to every classifier for class 1.

$$g_1(x) = \sum_{l=1}^{L} w_{(l,1)} h_{(l,1)}(x) \tag{1.2}$$

In case of simple averaging $w_{(l,1)} = \frac{1}{L}$. The performance of the simple average is often close to that of the weighted average (Fumera and Roli, 2005). It has however, been proven that the simple average is the optimal weighting only when the individual classifiers exhibit identical error rates and identical pair-wise correlations between the estimation errors (Fumera and Roli, 2005). Otherwise, a weighted average is always able to outperform it.

In this dissertation three classes of linear weight estimation methods are considered: the authority- based method, specialized optimization methods (i.e., traditional numerical methods such as logistic regression), and general purpose optimization methods (i.e., heuristics). For the latter method both population- based methods (e.g., genetic algorithms), and single solution or trajectory- based methods (e.g., simulated annealing) are considered.

In authority- based weight estimation the weight $w_{l,1}$ of each classifier l for class 1 is set proportional to its performance α on a validation set (Opitz and Shavlik, 1996).

$$w_{(l,1)} = \frac{\alpha_i}{\sum_{l=1}^{l} (\alpha_l)} \tag{1.3}$$

When specialized optimization methods and general purpose optimization methods (see Equation 1.2) are used an extra model is learned on top of the base classifiers. The first layer includes the base classifiers and the second layer is stacked on top of the first layer.

For the sake of completeness[2], in case of a mixture of experts, gating is used instead of stacking. The key difference with stacking is that weight estimation also incorporates the input data. Gating is defined as follows:

$$g_1(x) = \sum_{l=1}^{L} w_{(l,1)}(x) h_{(l,1)}(x) \tag{1.4}$$

This dissertation contains three studies in the field of ensemble learning. The first study introduces a novel method called Kernel Factory and validates the method on multiple data sets. The second study is an application study and benchmarks Kernel Factory in the domain of Customer Relationship Management. The third study proposes Hybrid Ensemble along with applications to several distinct problems. Table 1.1 summarizes the design elements of Kernel Factory and Hybrid Ensemble.

The following subsection leans on the current section to cover the research objectives of this dissertation. The penultimate section provides the main findings and the final section discusses limitations and directions for future research.

[2]This dissertation focuses on the simple average, authority- based weighting and stacking

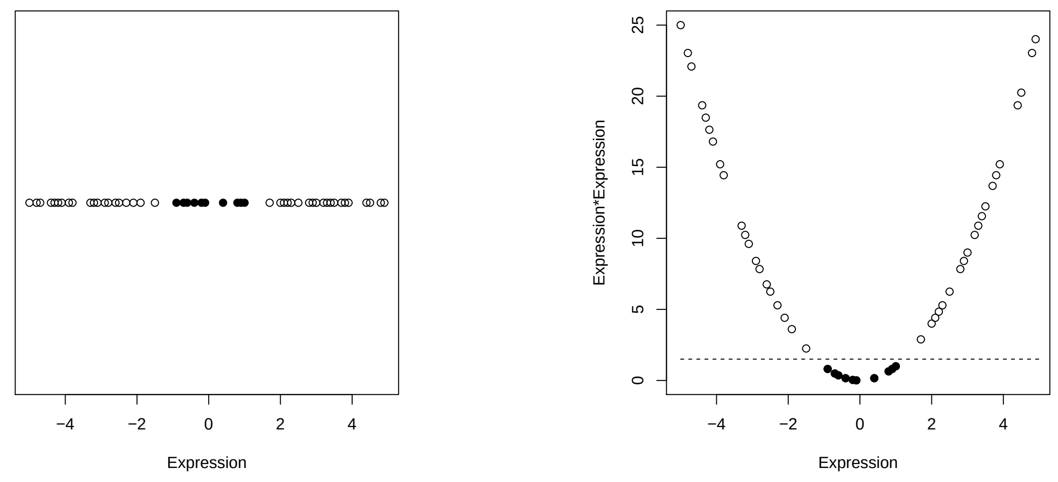
Table 1.1: Link to literature of the studies Kernel Factory and Hybrid Ensemble

Design phase	Characteristic	Study: Kernel Factory	Study: Hybrid Ensemble
Motivation	Main focus	Computational	Representational
Generation	Strength base classifiers	Strong	Strong
	Samples	Disjoint	Intersecting
	Ensemble homogeneity	Homogeneous	Heterogeneous
	Diversity strategy	Data	Data and algorithm
Combination	Weighting method	Stacking	Simple average, Authority, Stacking

1.2 Research Objectives

1.2.1 Study 1: Kernel Factory

The main objective of this study is to assess whether a divide- and- conquer approach can alleviate the computational problems of kernels. First, consider the main motivation for using kernels. The left panel in Figure 1.2 (adapted from Noble, 2006) illustrates a one-dimensional data set. The problem is that the dots and circles are inseparable by a single point. By adding an additional dimension to the data (in this case squaring the original expression) the circles and dots become separable by a straight line in the two-dimensional space.



Inseparable one-dimensional data set
Separable data set

Figure 1.2: Motivation for using kernels

Despite the clear advantages of this approach, analysis may soon become intractable. If there are 100 features in the data set instead of one, the resulting data set will contain 5050 features. The computational effort for a subsequent learner scales with the dimensionality (i.e., number of features n) of the new higher dimensional space, also called linearization (Jäkel et al., 2007) or feature space $\mathcal{F}$. Kernels offer a solution to this feature explosion so that the computational effort scales linearly with the size (i.e., number of observations N) of the original space, called input space $\mathcal{X}$ (Schölkopf and Smola, 2002).

Consider an n dimensional input space $\mathcal{X}$ and a mapping function ϕ that transforms $\mathcal{X}$ to $\mathcal{F}$. The function ϕ creates all n^d possible ordered[3] monomials of degree $d = 3$. For the mapping $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^{n^d}$, and two instances x_i and x_j, it holds that (Jäkel et al., 2007):

$$\begin{aligned}\langle\phi(x_i), \phi(x_j)\rangle &= \sum_{p_1=1}^{n}\sum_{p_2=1}^{n}\cdots\sum_{p_d=1}^{n} x_{ip_1}x_{ip_2}\ldots x_{ip_d}x_{jp_1}x_{jp_2}\ldots x_{jp_d} \\ &= \sum_{p_1=1}^{n} x_{ip_1}x_{jp_1}\sum_{p_2=1}^{n} x_{ip_2}x_{jp_2}\ldots\sum_{p_d=1}^{n} x_{ip_d}x_{jp_d} \\ &= \left(\sum_{p=1}^{n} x_{ip}x_{jp}\right)^d = \langle x_i, x_j\rangle^d \qquad (1.5)\end{aligned}$$

Equation 1.5 shows that the dot product in $\mathcal{F}$ equals the dot product in $\mathcal{X}$ to the power of d. It is unnecessary to map x_i and x_j to n^d dimensional feature space to calculate the inner product (Jäkel et al., 2007). The function $\langle\phi(x_i), \phi(x_j)\rangle = \langle x_i, x_j\rangle^d$ is called a kernel, short for kernel function, and is denoted by $k(x_i, x_j)$. The kernel in Equation 1.5 is a polynomial kernel.

A kernel matrix K, also called Gram matrix, is a matrix that collects all pairwise applications of a kernel function $k(.,.)$ to a set of instances x_1 to x_N. K is an N x N matrix

$$K = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \cdots & k(x_N, x_N) \end{pmatrix}$$

with an entry in the ith row and jth column denoted as k_{ij}:

$$k_{ij} = k(x_i, x_j) \qquad (1.6)$$

Kernels provide a novel data representation in that data are not represented individually anymore, but through a set of pairwise comparisons or similarities (Vert et al., 2004). A kernel matrix acts as an information bottleneck, since all the information available to a kernel learner (e.g., about distribution, model, noise) must be extracted from this matrix (Shawe-Taylor and Cristianini, 2004, p47). It is perhaps surprising how much information about an input data set can be obtained simply from its kernel matrix (Shawe-Taylor and Cristianini, 2004, p138).

Kernels have been introduced to mainstream machine learning literature in 1992 with the paper of Boser et al. (1992) on support vector machines. While for some time after that paper kernels were only used with large margin algorithms, the idea emerged that other types of algorithms could also be used in this way (Shawe-Taylor and Cristianini, 2004, p138). Any

[3]The order does not matter because of the commutativity of the product, but for simplicity we take the ordered monomials.

algorithm that can be kernelized (i.e., its dependence on the data is only through dot products) can be used with kernels (e.g., kernel logistic regression, parzen windows) (Ben-Hur and Weston, 2010, p4). Moreover, more recently it has been shown that training algorithms need not be kernelizable to be used with kernel matrices (Balcan et al., 2006, 2008; Fan, 2009). Fan (2011) professes that any algorithm can be used. Fan (2009); Fan et al. (2010); Cao and Fan (2010) show that, similar to tree procedures in input space, one can recursively partition the data space using kernel-induced features and aim to obtain homogeneous distributions of the response variable in the terminal nodes of the tree. The kernel- induced features are inherently linked to instances. For example, the fifth column in the kernel matrix, $k(., x_5)$, is constructed by, at each row taking the inner product of the respective row with the fifth row of the input space. In other words, the fifth column contains the degree of similarity of row five with all rows. This is what Fan (2009) calls a kernel-induced feature and x_5 is what Balcan et al. (2008, p91) call a landmark data point. The combination of kernels and trees, or a forest of trees, is called Kernel-Induced Classification Trees and Kernel-Induced Random Forest (Fan, 2009).

One of the main advantages of using kernel functions is that the resulting kernel matrix is always N x N, regardless of the complexity of the instances. This is very attractive when the number of instances is smaller than the number of features that would have been created if they were computed explicitly. For example, a data set with 5000 instances and 150 dimensions, and a polynomial function of degree 2 would result in 11325 dimensions. Using a kernel the dimensionality would only be 5000. Kernels become even more attractive when a relatively small number of complex objects needs to be processed (Vert et al., 2004).

Although kernels can reduce the computational effort from scaling with the number of features to scaling with the number of instances, the data set can still grow too large to analyze. To alleviate the problem Study 1 introduces a divide-and-conquer approach. Instead of computing K on the entire input data set, data are randomly split into equally sized disjoint sub-samples. Each sample is subsequently transformed by a kernel function into a kernel matrix K. Next each K is used as training data for a Random Forest. The predictions are then combined using a weighted average. The weights are computed using a genetic algorithm. This approach reduces memory requirements as the maximum used memory is now bounded by N of the sub-samples instead of N of the entire input data set (the latter being the case in Kernel-Induced Random Forest). The main research question is then whether the predictive performance of this approach can match the performance of Kernel-Induced Random Forest.

1.2.2 Study 2: CRM in Social Media

The purpose of this study is to assess the feasibility of predicting increases in Facebook usage frequency and to benchmark the performance of Kernel Factory in a Customer Analytics context. Kernel Factory is compared to Logistic regression, Random Forest, Stochastic Boosting, Support Vector Machines and a feed-forward Neural Network.

This study also touches upon the issue of interpretability. While ensemble methods are often credited for creating powerful predictive models, they are discredited for having low interpretability (see Figure 1.3; adapted from Gareth, Witten, Hastie, and Tibshirani 2013) (De Bock and Van den Poel, 2012; Gareth et al., 2013). Neslin et al. (2006) clearly denote that different algorithms are often required depending on whether the goal is description or prediction. Gareth et al. (2013) proclaim that as flexibility of a method increases, its interpretability decreases. In Gareth et al. (2013) flexibility is synonymous to non-linearity which is a proxy for predictive performance. It is important to note that it might not always pay off to select the most flexible method because more flexibility can translate into a higher potential for overfitting (Gareth et al., 2013, p26).

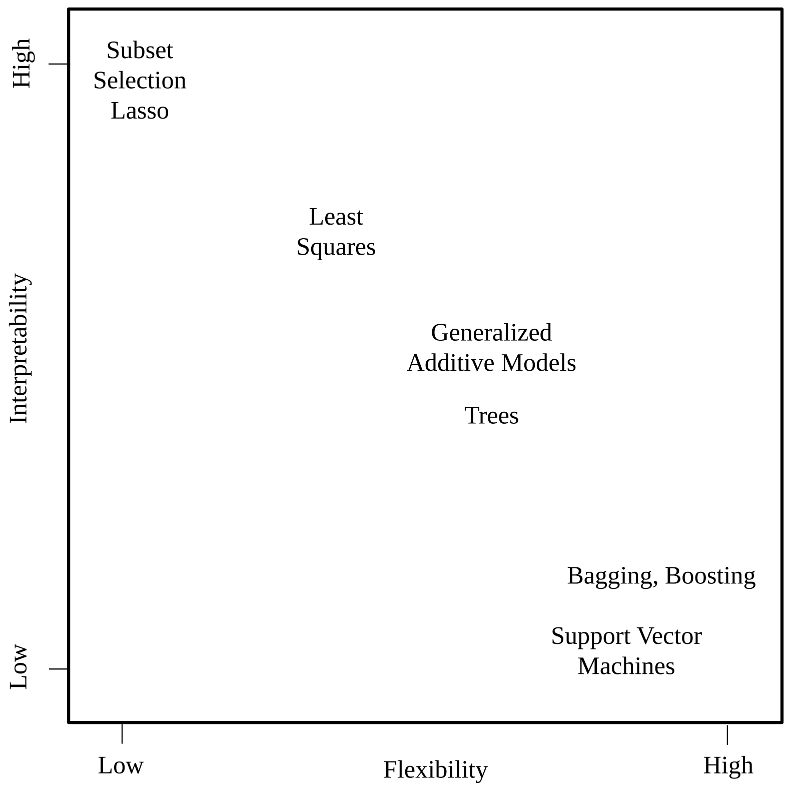


Figure 1.3: Trade-off between flexibility and interpretability

Because ensemble methods have low interpretability they are often not considered in fields such as Business Analytics. Model interpretability is advocated by several authors to be one of the main requirements for a successful model (Qi et al., 2009; De Bock and Van den Poel, 2012). In Customer Relationship Management, interpretable and intuitive models allow decision makers to gather valuable insights into the drivers of customer behavior (Masand et al., 1999). While it is common knowledge that the importance of predictors can easily be determined using permutation based measures (Breiman, 2001), the criticisms surrounding ensembles about interpretation mainly focus on their incapability of describing the sign or form of the relationship between a predictor and response.

There is however, a method for describing the relationship between a predictive feature and the response (in this case the probability to increase Facebook usage) using an ensemble model. This method is based on visualization: one of the most informative interpretational tools (Friedman and Meulman, 2003, p24). Partial Dependence Plots (Friedman, 2001; Hastie et al., 2009, Section 10.13.2) can describe the effects of features on the predictions of any black box learning method (Hastie et al., 2009; Cutler et al., 2007).

In general, a classification function h depends on many predictors $x = (x_1, \ldots, x_n)$. Consider a subset x_s containing $p < n$ of those predictors that are of interest to the researcher. The predictors x_c are the complement such that $x = x_s \cup x_c$ and $h(x) = h(x_s, x_c)$. The partial or average dependence of $h(x)$ on x_s is then defined by (Hastie et al., 2009)

$$f_s(x_s) = \frac{1}{N} \sum_{i=1}^{N} h(x_s, x_{ic}) \tag{1.7}$$

where $\{x_{1C}, x_{2C}, \ldots, x_{NC}\}$ are the values of x_c that occur in the training data. This method requires a pass over the data for each set of joint values of x_s (Hastie et al., 2009).

In classification tasks, $h(x)$ is computed on a logit scale. According to Liaw (2014) the underlying reason is that that is where the predictions are closer to symmetric (normal) and homoscedastic, where computing the mean makes sense. Let $P_k(x)$ be the probability of membership of the k^{th} class and N the number of instances. For K-classification the k^{th} response function is then given by Equation 1.8 (Hastie et al., 2009). In case of binary classification Equation 1.8 can be reduced to Equation 1.9.

$$f_s(x_s) = \frac{1}{N} \sum_{i=1}^{N} \left(log\big(P_k(x_s, x_{ic})\big) - \frac{1}{K} \sum_{k=1}^{K} log\big(P_k(x_s, x_{ic})\big) \right) \tag{1.8}$$

$$\begin{aligned} f_s(x_s) &= \frac{1}{N} \sum_{i=1}^{N} \left(log\big(P(x_s, x_{ic})\big) - \frac{1}{2}\Big(log\big(P(x_s, x_{ic})\big) + log\big(1 - P(x_s, x_{ic})\big) \Big) \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \Big(log\big(P(x_s, x_{ic})\big) - log\big(1 - P(x_s, x_{ic})\big) \Big) \\ &= \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \Big(logit\big(P(x_s, x_{ic})\big) \Big) \end{aligned} \tag{1.9}$$

Thus, the scale in Partial Dependence Plots is half of the logit (log of the odds) of the probability of the event. Please note that Partial Dependence Plot do not follow the convention to take one class as reference category (Berk, 2008, p224). While the overall fit is independent of the choice of reference category, coefficients are affected of by choosing one reference category over the other. Since there is no statistical justification for choosing a particular reference category, and the choice is mostly made on subject matter grounds, the choice varies from researcher to researcher. To avoid these complications, Partial Dependence Plots take the simple mean of the predictions of the K categories as reference. The response variable units are then deviations from the mean or the disparity between the logged predictions for category k and the mean of the logged predictions for all K categories. The units are logits but with the mean over the K categories as reference. Each category has its own equation and its own Partial Dependence Plot. This convention is used even when $K = 2$ and the conventional logit may not result in interpretation issues (Berk, 2008, p224).

From a more practical perspective, Partial Dependence Plots (for ensemble models) are created by the following procedure (Berk, 2008, p222):

1. Grow an ensemble model
2. For a specific predictor x_p, create a Partial Dependence Plot as follows:
   (a) Let v be the number of distinct values of a predictor x_p
   (b) For each value of the v values
      i. Create a novel data set where x_p only takes on that value and leave all other variables untouched
      ii. Predict the response for all instances in that novel data set using the ensemble
      iii. Calculate the mean of the half logit of the predictions yielding one single value for all instances
   (c) Plot the mean of half the logit of the predictions against the v values
3. Return to step 2 and repeat for other predictors of interest

It is important to note that Partial Dependence Plots represent the effect of a predictor or a subset of predictors after accounting for the effects of the other predictors. Analogously to a multiple linear regression, the coefficient of predictor x_1, obtained from regressing y on all x_j, measures the effect of x_1 accounting for the effects of the other variables. In simple linear regression where each x_j is regressed separately on y the coefficient of x_1 ignores the other predictors (Friedman and Meulman, 2003).

Study 2 will use Partial Dependence Plots for interpreting the effect of selected predictors on the probability of increasing Facebook usage frequency. The objective is to illustrate that ensemble methods should not be discredited for having interpretation issues.

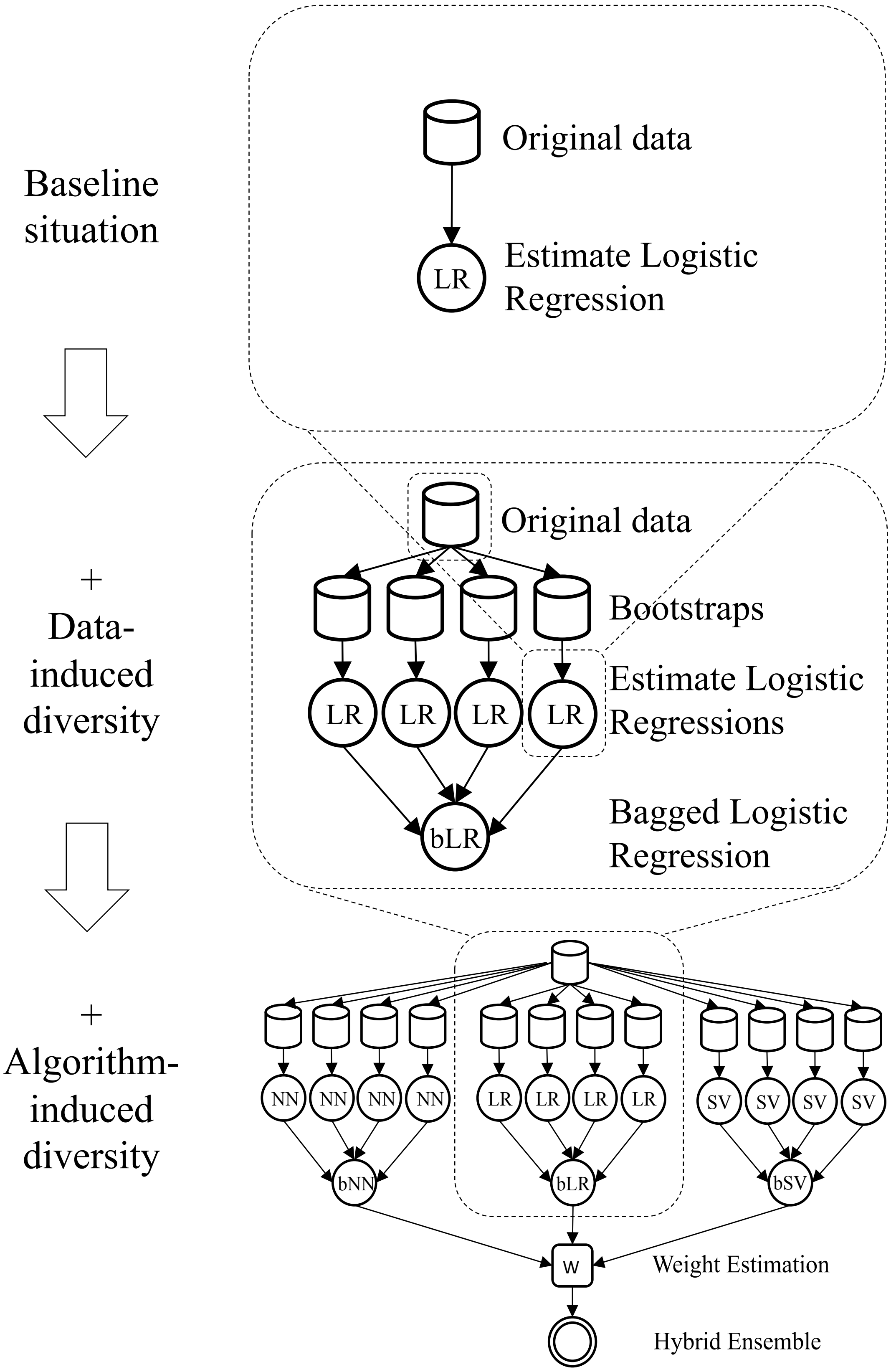
1.2.3 Study 3: Hybrid Ensemble

One of the most popular strategies for increasing classification performance is introducing data-induced diversity (see the top and middle panels in Figure 1.4). As mentioned in Section 1.1.2, key examples are Bagging (Breiman, 1996a), Boosting (Freund and Schapire, 1996) and Random Subspaces (Ho, 1998). Next to their effectiveness in improving classification, data- strategies thank their popularity to their ease of implementation. For example, Bagging consists in simply looping over a process of bootstrap sampling and classifier estimation, and aggregating the results. Algorithms that benefit the most from data-driven ensembling are unstable algorithms such as trees (Breiman, 1996a). Nevertheless, this strategy is also effective when more stable algorithms such as Neural Networks (Zhou et al., 2002), Logistic Regression (Kim, 2006), or Support Vector Machines (Valentini et al., 2003) are used.

Even when a given (ensemble) learner outperforms other learners on many tasks, as is the case with Breiman's (2001) Random Forest (Zhou, 2012, p21), one should still benchmark algorithms on each new task. There is no such thing as a free lunch in that no (homogeneous ensemble) algorithm performs best on all possible data sets (Wolpert, 1996). The strategy of evaluating multiple algorithms and selecting the best performing one is called the Single Best. Study 2 uses this strategy. It is one of the simplest yet most reliable and hence industrially preferred strategies (Ruta and Gabrys, 2005). It is important to note however, that the Single Best is considerably more complex than simply picking an algorithm (that one is comfortable with) and applying a data-driven ensemble technique. It requires a large amount of experience to be able to correctly apply certain algorithms. King et al. (1995) proclaim that many algorithms need an expert for parameter tuning. Tuning is of lesser concern in tree algorithms but is problematic in algorithms such as Neural Networks to the point of ”frustrating” (Ripley, 1993, p102) its users. It is the fact that expert knowledge is required for *all* the algorithms in the benchmark that drives the complexity of the Single Best method. Despite the difficulties inherent to implementing the Single Best method, it is the preferred option for improving classification performance in industrial applications (Ruta and Gabrys, 2005).

Although the Single Best is an effective method, optimal performance is not guaranteed (Roli and Giacinto, 2002, p208). There is a possibility that a combination of classifiers in a multiple classifier system (MCS) outperforms the best classifier (Ruta and Gabrys, 2005). This discovery came as a surprise to researchers from experiments in the late 1980s and early 1990s (Ho, 2002, p177). To ensure optimal performance, a MCS should be able to construct an optimal combination rule, adding an extra layer of complexity.

This study investigates whether introducing algorithm- induced diversity is a viable strategy over and above introducing data-induced diversity (see the bottom panel in Figure 1.4). It might well be the case that the latter strategy captures so much diversity that introducing additional algorithms does not increase predictive performance. If so, it is clear that this would make an algorithm variation strategy obsolete given the additional computational cost of a MCS in the estimation (i.e., the combination rule) and deployment phase.



(SV= Support Vector Machines, NN= Neural Network)

Figure 1.4: Selected components of the proposed Hybrid Ensemble.

1.3 Main findings, contributions and implications

Table 1.2 contains an overview of the main findings, along with contributions to literature and practical implications. In the following paragraphs the findings of the three studies are discussed in more detail.

Study 1. Using five times twofold cross- validation Kernel Factory is benchmarked against Random Forest and Kernel-Induced Random Forest (KIRF). The findings are fourfold. First, we found that one column partition works better than many. The training data are randomly split into a number of mutually exclusive partitions defined by a row and column parameter. Each partition forms an input space and is transformed by a kernel function into a kernel matrix K. Subsequently, each K is used as training data for a base binary classifier (Random Forest). While partitioning the columns is primarily a way of introducing diversity in the ensemble, the results indicate that one partition works best. Therefore more than one column partition is only recommended in case of numerical problems when computing the Ks (which is often the case in data sets with many features). Second, in addition to its superior speed on large data sets, Kernel Factory is significantly better than Kernel-Induced Random Forest (KIRF) on several data sets (and performs rarely significantly worse than KIRF). Third, the two methods (random and burn-in) that automatically select the kernel function perform equally well and using them is a viable strategy when the right kernel function is unknown in advance. Fourth and final, when using a kernel is appropriate, and the right kernel is specified, both Kernel Factory and Kernel-Induced Random Forest outperform Random Forest significantly.

Study 2. The results clearly indicate that usage increase prediction is a viable strategy with five times twofold cross-validated AUCs up to 0.66 and accuracies up to 0.74. When AUC is used as performance measure Stochastic Boosting is the best choice to model usage increase, followed by Random Forest, Logistic Regression, Kernel Factory, Support Vector Machines and Neural Networks. When using accuracy, the first position also goes to Stochastic Boosting. Random Forest and Support Vector Machines share the second position, Kernel factory and Logistic regression share the third place and Neural Network performs worst. The top predictor is the time ratio. This variable is operationalized to capture the user's deviation from his or her regular usage patterns. The Partial Dependence Plot indicates a strong negative relationship. Greater deviations from regular usage patterns results in lower probability of usage increase. Other important predictors include frequencies of likes of specific categories and group memberships, average photo album privacy settings, and recency of comments.

The fact that Kernel Factory only obtains an intermediate position in this benchmark has given rise to the basic idea of MCSs underlying Study 3 (Hybrid Ensemble). In fact, Kernel Factory is heavily dependent upon the choice of the kernel function (as any kernel-based method for that matter). The latter is in turn contingent upon the training data. Using a kernel, as opposed to not using a kernel, is not always the right choice. This makes Kernel Factory not very versatile. Study 3 therefore aims to create an algorithm that can be effectively employed in a wide spectrum of tasks and applications.

Study 3. In this study a new Hybrid Ensemble is proposed consisting of six sub-ensembles: Bagged Logistic Regression, Random Forest, Kernel Factory, Bagged Support Vector Machines, Stochastic Boosting, and Bagged Neural Networks. Using five times twofold cross-validation, we found that the proposed Hybrid Ensemble yields better performance than the Single Best on all tested data sets for the authority- based weight estimation method (i.e., weighting according to individual performance) and the Self-Organising Migrating Algorithm (SOMA) weight estimation method. The maximum incremental AUC that those methods bring over the Single Best is respectively 0.025 and 0.022, which is sizable given the effective data-strategy the Single Best uses. Since the authority- based method outperforms SOMA, and in addition is relatively computationally more efficient, we recommend it as the default option in future research. The sub-ensembles generate diversity through data perturbation. Hence the difference of the performance of the Hybrid Ensemble and the Single Best sub-ensemble effectively yields the added value of algorithm-induced diversity. It is also found that the added value of the Hybrid Ensemble over the Single Best increases with increasing classification difficulty of the task at hand. This study also gives an indication that six sub-ensembles is sufficient to constitute the Hybrid Ensemble. In addition, there is no clear winning or losing sub-ensemble. These findings indicate the appropriateness of the selected algorithms. The main implication is that the industrially preferred Single Best method should be replaced by Hybrid Ensembling.

1.4 Limitations and future research

Study 1. Kernel Factory was designed to improve upon Fan's (2009) Kernel-Induced Random Forest. It was not designed to compete with non-kernel based methods such as Random Forest. This design choice is also clearly reflected in the benchmark results. If the appropriate kernel cannot be found predictive performance will suffer (Üstün et al., 2006). If found, Kernel Factory can drastically improve upon Random Forest. In order to find the right kernel a process of cross-validation has to be undertaken. When evaluating Kernel Factory, Random Forest should always be benchmarked because there is no guarantee that Kernel Factory will be at least at good as Random Forest. This is the main disadvantage of the algorithm. For researchers that want an off-the-shelf algorithm that performs well on a wide array of tasks, Kernel Factory may not be a good choice. However, Kernel Factory is a good candidate in the Single Best method. Hence, a burn-in automatic kernel selection has been built into Kernel Factory in an attempt to increase versatility. However, future research may consider other ways to increase the versatility of Kernel Factory. An interesting avenue would be to include more generic or universal kernel functions (Üstün et al., 2006; Zhang and Wang, 2011). These functions are robust and have stronger mapping ability compared to the standard kernel functions, resulting in better generalization performance while reducing the amount of required tuning.

Another avenue for improving Kernel Factory is pruning. The main strength of Kernel

Table 1.2: *Contribution to literature, main findings, and practical implications*

Title	Contribution to literature	Main findings	Practical implications
Kernel Factory: An Ensemble of Kernel Machines (published in Expert Systems with Applications, 2013)	Assess effectiveness of divide-and-conquer technique when computing kernel matrix	Computing many kernel matrices on disjoint subsets of the input data does not incur a predictive performance penalty	• Speed improvements and increased computational efficiency thanks to possibility of parallelized and distributed execution • R-Package *kernelFactory*
CRM in Social Media: Predicting Increases in Facebook Usage Frequency (under review)	• Assess feasibility of prediction using social media data (Facebook) • Benchmark Kernel Factory against top performing algorithms in literature • Determine which variables are important • Illustrate how to interpret relationship between predictors and response using ensemble	• Usage frequency increase can be predicted with AUC=0.66 and accuracy=0.74 • Winner: Stochastic AdaBoost. Kernel Factory obtains 4^{th} position out of 6. • Important predictors include deviation from regular usage patterns, frequencies of likes of specific categories and group memberships, average photo album privacy settings, and recency of comments.	Social Media sites such as Facebook can use predictive modeling to pro-actively customize services such as pacing the rate of advertisements and friend recommendations.
Hybrid Ensembles: Many Ensembles is Better Than One (under review)	Assess added value of algorithm-induced diversity over data- induced diversity	• Hybrid Ensemble significantly and consistently outperforms Single Best • Greater improvements with more difficult tasks • Six team members is sufficient	• The industrially preferred Single Best method should be replaced by Hybrid Ensembling • R- Package *hybridEnsemble*

Factory is the resulting lower computational footprint. Currently all ensemble members are used and weighted in the prediction phase. By excluding members with weight close to zero Kernel Factory can be even less memory demanding and less computationally expensive in the prediction phase. Zhou et al. (2002) demonstrate that this principle works quite well. Future research could investigate the effectiveness and implication on generalization performance of this technique in Kernel Factory.

Study 2. This study focuses on estimating predictive (ensemble) models and shows how to interpret them using variable importance measures and Partial Dependence Plots. Future research might consider adding an additional layer in the form of a prescriptive model. The latter could be used to highlight the importance of interpretation tools in ensemble learning. For example, consider the task of optimizing Facebook network size. The following steps would be required: (1) estimate an ensemble model, (2) predict network size, (3) select the most important variables, (4) find optimal values for these variables, and finally (5) use partial dependence plots to verify their optimality. Step 3 and 5 would more clearly illustrate the value and usability of the interpretation tools that can be used with ensemble algorithms.

A second direction for improving this study is to compare Partial Dependence Plots to the inherent interpretation mechanism in other techiques such as coefficients in logistic regression or simple decision trees. An especially interesting idea for future research is to compare Partial Dependence Plots to rule extraction techniques for ensemble models (e.g., Al Iqbal, 2012).

Study 3. The focus of this study is on the ensemble generation phase. An interesting direction for future research is to assess the added value of algorithm-induced diversity over data-induced diversity while evaluating more flexible combination rules. This study uses the simplest and most widely used implementation of a weighted average, that is, when a nonnegative weight is assigned to each member (Tumer and Ghosh, 1996; Verikas et al., 1999; Fumera and Roli, 2005). This makes our method suitable for applications where the weights are intended as probabilities that a classifier gives a correct answer or where the final linear combination is intended to be interpreted as a probability estimate (Fumera and Roli, 2005). However Tumer and Ghosh (1996) argued that this method might not be sufficiently more flexible than the simple average. In response more flexible implementations have been proposed such as using a set of weights with no sign restrictions, using nonlinear weights, and even using different weights per member and per class (Benediktsson et al., 1997; Ueda, 2000). Nevertheless, there are some problems that result from the use of these flexible methods. It has been observed that weights unrestricted in sign can sometimes be difficult to estimate (Breiman, 1996b). Restricting the weights to be nonnegative has been suggested as a regularization method to avoid this problem (Breiman, 1996b; LeBlanc and Tibshirani, 1996). Nonlinear weights have the disadvantage that more parameters need to be estimated which can increase the required computational effort to an infeasible level. Estimating a weight per member and per class also has a disadvantage in that a greater quantity of validation data is required (Fumera and Roli, 2005). In sum, in this study the most widely used implementation of the weighted average was used. Future research might investigate whether the results hold when more flexible weighting methods are considered.

The second direction for future research is to extend the experiments to a larger number of data sets and analyze their characteristics. This may provide insight into which data characteristics dictate the use of which combination method. An especially interesting avenue is to consider larger and more difficult data sets.

The third direction is to study the optimality of the number of candidate base classifiers. Although this topic is briefly touched upon in this study, further research is needed to identify the possible gains related to increasing the number of team members. If those gains are sufficiently large, that may be a reason to increase the ensemble size. However, there are four reasons not to increase the ensemble size. First, including more team members would increase learning times and computational effort for both the base classifiers (Margineantu and Dietterich, 1997) and the combination methods to an impractical level. Second, small ensembles have, in many cases, near-optimal performance (Margineantu and Dietterich, 1997). Third, diversity has a strong effect in the ensemble performance when using less than ten classifiers (Kuncheva, 2004; Canuto et al., 2007). Fourth, we included only the top performing ensembles found in extant literature, given that the purpose of this paper is to improve upon the industrially preferred Single Best method. Future research may also consider to decrease the ensemble size and incorporate some form of classifier selection in addition to the classifier fusion step. This would involve evaluating ensembles of different sizes. However we do not expect this to yield significant improvement since zero-weights are allowed in the fusion step.

The fourth and undeniably most important direction for future research is to make an extensive analysis of diversity. Ideally the diversity of each of the six sub-ensembles and their combination (at the Hybrid Ensemble level) is compared using several measures of diversity. Two main obstacles need to be overcome. Some algorithms such as Random Forest produce a label per tree in the forest and per instance. At the ensemble level, Random Forest produces a probability (we choose probabilities for all sub-ensembles because it contains more information than labels). This means that two different diversity measures are required to compare the amount of data-induced diversity at the sub-ensemble level and the amount of algorithm-induced diversity at the meta-ensemble level. Because the outputs are incommensurate, either the algorithm and software needs to be changed or a diversity measure needs to be found that is completely unaffected by the nature of the outputs (i.e., labels or probabilities). As to the first option, some of the sub-ensembles (e.g., Random Forest, Stochastic Boosting) are implemented in packages (as is custom *R*). While the interface is in the *R*- language some parts of the code are implemented in lower level languages such as *Fortran* or *C++*. This makes it challenging to adapt the algorithm and code. As to the second option, to the best of our knowledge there exists no such diversity measure. A somewhat less challenging task would be to fit the relationship between the amount of algorithm-induced diversity and meta-ensemble performance across data sets. Given that all sub-ensembles produce probabilities the aforementioned problem is non-existent. However, in the light of extensive experimental evidence (see Ruta and

Gabrys (2005) for an overview) showing a very low correlation between diversity measures and ensemble performance, we do not expect this to be a fruitful undertaking.

A final avenue is to study the optimality of the members. In this study the goal was to compare the Hybrid Ensemble's performance with the Single Best. Hence the most popular and powerful algorithms were employed as base classifiers. Future research might study whether other, maybe less powerful members, can maybe result in higher meta-ensemble classification performance. In this light it is also important to investigate the interaction between the degree of diversity and weighting method. Fumera and Roli (2005) have shown that the performance improvement achievable by the weighted average, over the simple average, actually decreases when diversity is higher. Despite the limitation of that study that only unbiased estimation errors are considered, which is likely to be violated in real cases (Fumera and Roli, 2005), and the fact that only tests have been conducted with three base classifiers, this is an interesting finding to investigate further. This would mean that weight estimation can be eliminated in those cases. Although the current study focuses on the classifier generation phase and not on the classifier combination phase, an in depth investigation of the interaction of those two phases may provide valuable insights.

1.5 References

Al-Ani, M., Deriche, M., 2002. A new technique for combining multiple classifiers using the dempster-shafer theory of evidence. Journal of Artificial Intelligence Research 17, 333–361.

Al Iqbal, M. R., 2012. Rule extraction from ensemble methods using aggregated decision trees. Neural Information Processing. 19th International Conference (ICONIP 2012). Proceedings, 599–607.

Balcan, M.-F., Blum, A., Srebro, N., Aug. 2008. A theory of learning with similarity functions. Machine Learning 72 (1-2), 89–112.

Balcan, M.-F., Blum, A., Vempala, S., Oct. 2006. Kernels as features: On kernels, margins, and low-dimensional mappings. Machine Learning 65 (1), 79–94.

Bauer, E., Kohavi, R., Jul. 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36 (1-2), 105–139.

Ben-Hur, A., Weston, J., 2010. A user's guide to support vector machines. Methods in Molecular Biology. Department of Computer Science Colorado State University, pp. 223–239.

Benediktsson, J. A., Sveinsson, J. R., Ersoy, O. K., Swain, P. H., Jan. 1997. Parallel consensual neural networks. IEEE Transactions on Neural Networks 8 (1), 54–64.

Berk, R. A., 2008. Statistical Learning from a Regression Perspective. Springer Series in Statistics. Springer.

Boser, B. E., Guyon, I. M., Vapnik, V. N., 1992. A training algorithm for optimal margin classifiers. Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, 144–52.

Breiman, L., Aug. 1996a. Bagging predictors. Machine Learning 24 (2), 123–140.

Breiman, L., Jul. 1996b. Stacked regressions. Machine Learning 24 (1), 49–64.

Breiman, L., Oct. 2001. Random forests. Machine Learning 45 (1), 5–32.

Bryll, R., Gutierrez-Osuna, R., Quek, F., Jun. 2003. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognition 36 (6), 1291–1302.

Canuto, A. M. P., Abreu, M. C. C., Oliveira, L. d. M., Xavier, J. C., Santos, A. d. M., Mar. 2007. Investigating the influence of the choice of the ensemble members in accuracy and diversity of selection-based and fusion-based methods for ensembles. Pattern Recognition Letters 28 (4), 472–486.

Cao, J., Fan, G., 2010. Signal classification using random forest with kernels. Proceedings Sixth Advanced International Conference on Telecommunications (AICT 2010), 191–5.

Cutler, D. R., Edwards, T. C., Beard, K. H., Cutler, A., Hess, K. T., Nov. 2007. Random forests for classification in ecology. Ecology 88 (11), 2783–2792.

De Bock, K. W., Van den Poel, D., Jun. 2012. Reconciling performance and interpretability in customer churn prediction using ensemble learning based on generalized additive models. Expert Systems with Applications 39 (8), 6816–6826.

Dietterich, T. G., 2000. Ensemble methods in machine learning. In: Kittler, J., Roli, F. (Eds.), Multiple Classifier Systems. Vol. 1857. pp. 1–15.

Domingos, P., Oct. 2012. A few useful things to know about machine learning. Communications of the Acm 55 (10), 78–87.

Duin, R. P. W., 2002. The combining classifier: to train or not to train? In: Kasturi, R., Laurendeau, D., Suen, C. (Eds.), 16th International Conference on Pattern Recognition, Vol Ii, Proceedings. IEEE Computer Soc, Los Alamitos, pp. 765–770.

Fan, G., 2009. Kernel-induced classification tree and random forest. Technical report, Dept. of Statistics and Actuarial Science, University of Waterloo.

Fan, G., 2011. personal communication.

Fan, G., Wang, Z., Kim, S. B., Temiyasathit, C., 2010. Classification of high-resolution NMR spectra based on complex wavelet domain feature selection and kernel-induced random forest. In: Elmoataz, A., Lezoray, O., Nouboud, F., Mammass, D., Meunier, J. (Eds.), Image and Signal Processing, Proceedings. Vol. 6134. Springer-Verlag Berlin, Berlin, pp. 593–600.

Freund, Y., Sep. 1995. Boosting a weak learning algorithm by majority. Information and Computation 121 (2), 256–285.

Freund, Y., Schapire, R., 1996. Experiments with a new boosting algorithm. In: Machine Learning. Proceedings of the Thirteenth International Conference (ICML ’96). Bari, Italy, pp. 148–156.

Friedman, J. H., Oct. 2001. Greedy function approximation: A gradient boosting machine. Annals of Statistics 29 (5), 1189–1232.

Friedman, J. H., Meulman, J. J., May 2003. Multiple additive regression trees with application in epidemiology. Statistics in Medicine 22 (9), 1365–1381.

Fumera, G., Roli, F., Jun. 2005. A theoretical and experimental analysis of linear combiners for multiple classifier systems. Ieee Transactions on Pattern Analysis and Machine Intelligence 27 (6), 942–956.

Gareth, J., Witten, D., Hastie, T., Tibshirani, R., 2013. An Introduction to Statistical Learning with application in R. Springer Texts in Statistics. Springer.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning - Data Mining, Inference, and Prediction, second edition Edition. Springer Series in Statistics. Springer.

Ho, T. K., Aug. 1998. The random subspace method for constructing decision forests. Ieee Transactions on Pattern Analysis and Machine Intelligence 20 (8), 832–844.

Ho, T. K., 2002. Multiple classifier combination: Lessons and next steps. In: Bunke, H., Kandel, A. (Eds.), Hybrid Methods in Pattern Recognition. Vol. 47 of Series in Machine Perception and Artificial Intelligence. World Scientific Publishing, pp. 146–198.

Jäkel, F., Schölkopf, B., Wichmann, F. A., Dec. 2007. A tutorial on kernel methods for categorization. Journal of Mathematical Psychology 51 (6), 343–358.

Kim, Y., 2006. Toward a successful CRM: variable selection, sampling, and ensemble. Decision Support Systems 41 (2), 542–553.

King, R., Feng, C., Sutherland, A., 1995. Statlog - comparison of classification algorithms on large real-world problems. Applied Artificial Intelligence 9 (3), 289–333.

Kuncheva, L., 2004. Combining Pattern Classifiers. Methods and Algorithms. Wiley.

LeBlanc, M., Tibshirani, R., Dec. 1996. Combining estimates in regression and classification. Journal of the American Statistical Association 91 (436), 1641–1650.

Liaw, A., 2014. personal communication.

Margineantu, D. D., Dietterich, T. G., 1997. Pruning adaptive boosting. In: Proc. 14th Int'l Conf. Machine Learning. pp. 211–218.

Masand, B., Datta, P., Mani, D. R., Li, B., Jun. 1999. CHAMP: a prototype for automated cellular churn prediction. Data Mining and Knowledge Discovery 3 (2), 219–225.

Neslin, S. A., Gupta, S., Kamakura, W., Junxiang Lu, Mason, C. H., May 2006. Defection detection: Measuring and understanding the predictive accuracy of customer churn models. Journal of Marketing Research (JMR) 43 (2), 204–211.

Noble, W. S., Dec. 2006. What is a support vector machine? Nature Biotechnology 24 (12), 1565–1567.

Opitz, D. W., Shavlik, J. W., 1996. Generating accurate and diverse members of a neural-network ensemble. In: Touretzky, D. S., Mozer, M. C., Hasselmo, M. E. (Eds.), Advances in Neural Information Processing Systems 8: Proceedings of the 1995 Conference. Vol. 8. M I T Press, Cambridge, pp. 535–541.

Parikh, D., Polikar, R., Apr. 2007. An ensemble-based incremental learning approach to data fusion. Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics 37 (2), 437–450.

Qi, J., Zhang, L., Liu, Y., Li, L., Zhou, Y., Shen, Y., Liang, L., Li, H., Apr. 2009. ADTreesLogit model for customer churn prediction. Annals of Operations Research 168 (1), 247–265.

Ripley, B., 1993. Statistical aspects of neural networks. In: Barndorff-Nielsen, O., Jensen, J., Kendall, W. (Eds.), Networks and Chaos - Statistical and Probabilistic Aspects. Monographs on Statistics and Applied Probability 50. Chapman & Hall, London, pp. 40–123.

Roli, F., Fumera, G., Kittler, J., 2002. Fixed and trained combiners for fusion of imbalanced pattern classifiers. Int Soc Information Fusion, Sunnyvale.

Roli, F., Giacinto, G., 2002. Design of multiple classifier systems. In: Bunke, H., Kandel, A. (Eds.), Hybrid Methods in Pattern Recognition. Vol. 47 of Series in Machine Perception and Artificial Intelligence. World Scientific Publishing, pp. 199–226.

Ruta, D., Gabrys, B., Mar. 2005. Classifier selection for majority voting. Information Fusion 6 (1), 63–81.

Schapire, R., Jun. 1990. The strength of weak learnability. Machine Learning 5 (2), 197–227.

Schölkopf, B., Smola, A., 2002. Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA.

Sharkey, A. J. C., Sharkey, N. E., Sep. 1997. Combining diverse neural nets. Knowledge Engineering Review 12 (3), 231–247.

Shawe-Taylor, J., Cristianini, N., 2004. Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge, UK.

Tsoumakas, G., Angelis, L., Vlahavas, I., 2005. Selective fusion of heterogeneous classifiers. Intelligent Data Analysis 9 (6), 511–525.

Tumer, K., Ghosh, J., Feb. 1996. Analysis of decision boundaries in linearly combined neural classifiers. Pattern Recognition 29 (2), 341–348.

Ueda, N., Feb. 2000. Optimal linear combination of neural networks for improving classification performance. Ieee Transactions on Pattern Analysis and Machine Intelligence 22 (2), 207–215.

Üstün, B., Melssen, W. J., Buydens, L. M. C., Mar. 2006. Facilitating the application of support vector regression by using a universal pearson VII function based kernel. Chemometrics and Intelligent Laboratory Systems 81 (1), 29–40.

Valentini, G., Muselli, M., Ruffino, F., 2003. Bagged ensembles of support vector machines for gene expression data analysis. In: Proceedings of the International Joint Conference on Neural Networks 2003, Vols 1-4. Ieee, New York, pp. 1844–1849.

Verikas, A., Lipnickas, A., Malmqvist, K., Bacauskiene, M., Gelzinis, A., Apr. 1999. Soft combination of neural classifiers: A comparative study. Pattern Recognition Letters 20 (4), 429–444.

Vert, J.-P., Tsuda, K., Schölkopf, B., 2004. A primer on kernel methods. In: Schölkopf, B., Tsuda, K., Vert, J.-P. (Eds.), Kernel Methods in Computational Biology. Computational Molecular Biology. MIT Press, pp. 35–70.

Windeatt, T., Mar. 2005. Diversity measures for multiple classifier system analysis and design. Information Fusion 6 (1), 21–36.

Wolpert, D. H., Oct. 1996. The lack of a priori distinctions between learning algorithms. Neural Computation 8 (7), 1341–1390.

Xu, L., Krzyzak, A., Suen, C., Jun. 1992. Methods of combining multiple classifiers and their applications to handwriting recognition. Ieee Transactions on Systems Man and Cybernetics 22 (3), 418–435.

Zhang, R., Wang, W., Oct. 2011. Facilitating the applications of support vector machine by using a new kernel. Expert Systems with Applications 38 (11), 14225–14230.

Zhou, Z.-H., 2012. Ensemble Methods: Foundations and Algorithms. Machine Learning & Pattern Recognition Series. Chapman & Hall/CRC, Boca Raton FL.

Zhou, Z. H., Wu, J. X., Tang, W., May 2002. Ensembling neural networks: Many could be better than all. Artificial Intelligence 137 (1-2), 239–263.

2

Kernel Factory: An Ensemble of Kernel Machines

Ballings, M, Van den Poel D. 2013. Kernel Factory: An ensemble of kernel machines. Expert Systems with Applications, 40(8), 2904-2913.

2.1 Abstract

We propose an ensemble method for kernel machines. The training data are randomly split into a number of mutually exclusive partitions defined by a row and column parameter. Each partition forms an input space and is transformed by a kernel function into a kernel matrix K. Subsequently, each K is used as training data for a binary base classifier (Random Forest). This results in a number of predictions equal to the number of partitions. A weighted average combines the predictions into one final prediction. To optimize the weights, a genetic algorithm is used. This approach has the advantage of simultaneously promoting (1) diversity, (2) accuracy, and (3) computational speed. (1) Diversity is fostered because the individual Ks are based on a subset of features and observations, (2) accuracy is sought by using strong base classifiers as opposed to stumps, and (3) computational speed is obtained because the computation of each K can be parallelized. Using five times two-fold cross- validation we find that Kernel Factory has significantly better performance than Kernel-Induced Random Forest. When the right kernel is specified Kernel Factory is also significantly better than Random Forest. In addition, we submitted an open-source R-software package of the algorithm ($kernelFactory$) to CRAN.

2.2 Introduction

In the last decade, kernel- based methods have become very popular for classification, regression and pattern recognition (Üstün et al., 2006). It has been shown that classifiers can be improved by mapping input space $\mathcal{X}$ into feature space $\mathcal{F}$ (Jäkel et al., 2007). However, this mapping can increase the number of dimensions substantially to the point that analysis becomes problematic. Kernel methods are attractive in that they can create the aforementioned mapping and temper the dimensionality explosion problem by increasing the number of dimensions only linearly with the size of the original data (Shawe-Taylor and Cristianini, 2004).

Despite this beneficial property of kernel methods, data can grow very large. Hence, researchers have to resort to machine learning techniques that can handle a large number of predictors, such as Random Forest. In that context, it is only recently that Kernel- Induced Random Forest (KIRF) has been introduced (Fan, 2009). The advantage of KIRF over support vector machines (SVM) is that the former can handle remaining non-linearities in $\mathcal{F}$. Nevertheless, data can grow so large that classifier accuracy suffers due to the lower probability of selecting informative features. It can even grow unwieldy making analysis infeasible.

In an attempt to alleviate these computational efficiency problems, we propose an ensemble of kernel machines: Kernel Factory. In addition, Kernel Factory has the advantages of increased computational speed and ensemble member diversity. The remainder of this article is organized as follows. Section 2 reviews Random Forest, the kernel trick, and KIRF. Section 3 describes the proposed method Kernel Factory in detail. In Section 4 we present the methodology and results of an empirical study in which we benchmark Kernel Factory against Random Forest and KIRF. Section 5 provides the discussion and conclusion of the results. Finally, Section 6 offers limitations and avenues for future research.

2.3 Kernels and Random Forest

In this section we first elaborate on Random Forest. Second, we discuss the attractiveness of kernels. Third, we discuss the combination of both: Kernel-Induced Random Forests.

2.3.1 Random Forest

Binary recursive partitioning (BRP) is a method that grows decision trees, also referred to as classification and regression trees (CART) (Breiman et al., 1984). The BRP algorithm starts by predicting a criterion variable by creating a binary partitioning of the data based on one predictor. The algorithm proceeds recursively by, within a parent partition, creating two child-partitions of the data. This partioning is based on another predictor or another split value of the same predictor variable that was used to create the parent partition (Merkle and Shaffer, 2011). At each partitioning step, the predictor that produces the purest division of data is selected and the algorithm stops when, for example, a minimum partition size, a specific impurity or an amount of partitions is reached.

BRP is also used in Random Forest (Breiman, 2001). Instead of growing one tree, Random Forest grows, and averages over, an ensemble of trees. Each tree is grown using an independent bootstrap sample for which at each partitioning step of a tree a subset of variables is randomly selected as splitting candidates (Breiman, 2001).

Literature shows that Random Forest is one of the best-performing classification techniques available (Luo et al., 2004). Moreover, it is very robust and consistent and does not overfit (Breiman, 2001). Furthermore, the algorithm has reasonable computing times (Buckinx and Van den Poel, 2005) and the procedure is easy to implement: only two parameters are to be set (number of trees and number of predictors) (Larivière and Van den Poel, 2005; Duda et al., 2001).

2.3.2 Kernels and the kernel trick

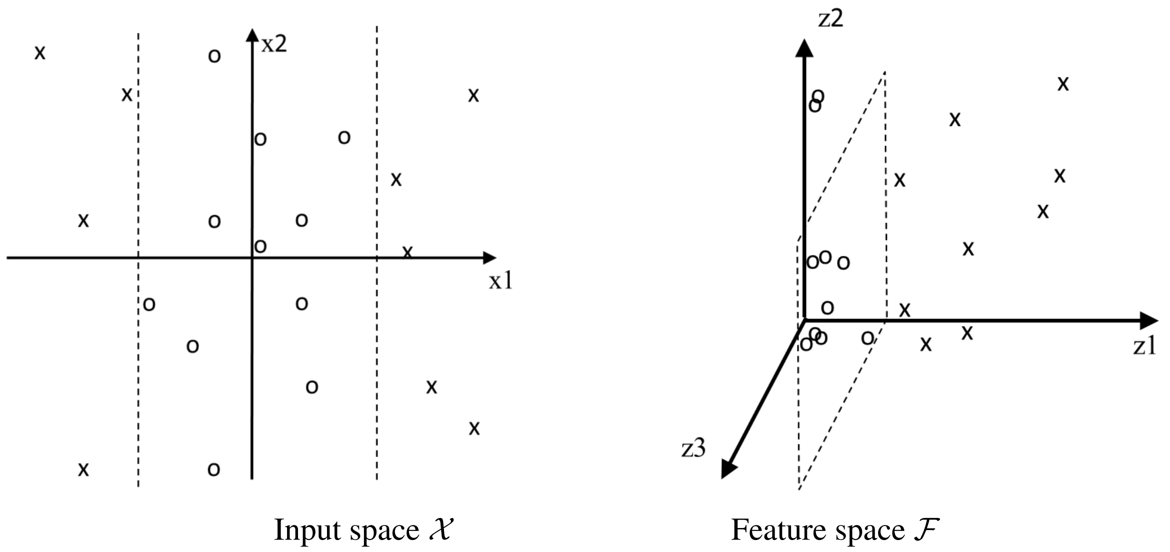
Consider the binary classification problem in the left pane of Figure 2.1(adapted from Schölkopf and Smola (2002)). A binary recursive partitioning tree would partition the data in two by using $x_1 >= 3$ as split rule in the root node and $x_1 <= 3$ in the child node. As such, to obtain a perfect classification the variable x_1 needs to be selected twice (x_2 has no discriminatory power in this case).

By applying feature map ϕ that transforms the input space (x_1, x_2), by taking all second-degree unordered monomials, a dimension can be found that reduces the tree size (see right pane of Figure 2.1; note that actually $z_3 = \sqrt{2}x_1x_2$ and not x_1x_2 but this will create a similar plot with the same decision boundary):

$$\begin{aligned}&\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3\\ &\phi(x_1, x_2) = (z_1, z_2, z_3) = (x_1^2, x_2^2, x_1x_2)\end{aligned} \tag{2.1}$$

In feature space, a binary recursive partitioning tree would partition the data by only one split, as opposed to two splits in input space, using $z1 >= 8$ as the split rule. Because of this reduction in tree size, the performance of Random Forest can be improved. In Random Forests, out of a the total set of predictor variables, at each node a subset of candidate variables is selected at random and the candidate that produces the best split is used to split the node (Breiman, 2001). Hence, because the probability is lower that the variable is selected twice as a candidate split variable, as opposed to once, the individual trees are bound to be stronger in feature space, as such decreasing the forest error rate (Breiman, 2001).

While working in feature space has its advantages, it clearly has its disadvantages as well. Although the classification performance is improved thanks to elimination of the need to select the variable in our example more than once, there are now more predictors, decreasing the



The feature map ϕ transforms the input space (x_1, x_2), by taking all second-degree unordered monomials, to the feature space (z_1, z_2, z_3) where only one binary split rule is required to obtain the same result.

Figure 2.1: Input space $\mathcal{X}$ and Feature space $\mathcal{F}$

probability of selection for the candidate set (it has to be noted that the size of the subset is dependent on the size of the total set).

Consider Equation 2.2 (Schölkopf and Smola, 2002) that shows that the dimensionality n in feature space can easily explode given monomials of degree d, making analysis prohibitive.

$$n_{\mathcal{F}} = \binom{d + n_{\mathcal{X}} - 1}{d} = \frac{(d + n_{\mathcal{X}} - 1)!}{d!(n_{\mathcal{X}} - 1)!} \tag{2.2}$$

where $\mathcal{X}$ denotes input space and $\mathcal{F}$ feature space

In our example $d = 2$ and $n_{\mathcal{X}} = 2$ which results in $n_{\mathcal{F}} = 3$. For $d = 2$ and $n_{\mathcal{X}} = 150$ dimensionality $n_{\mathcal{F}}$ amounts to 11,325. If in the latter case the degree increases by one ($d = 3$) then $n_{\mathcal{F}} = 573,800$. Calculating the inner product offers a solution by limiting the feature explosion so that the complexity of a classifier increases only linearly with the size of the input data. For two instances x_i and x_j (i,j=1,2,3,...,N) defined by two variables (x_1, x_2) (Jäkel et al., 2007):

$$\begin{aligned}\langle\phi(x_{i1}, x_{i2}), \phi(x_{j1}, x_{j2})\rangle &= \langle(x_{i1}^2, x_{i2}^2, \sqrt{2}x_{i1}x_{i2})(x_{j1}^2, x_{j2}^2, \sqrt{2}x_{j1}x_{j2})\rangle \\ &= x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} \\ &= (x_{i1}x_{j1} + x_{i2}x_{j2})^2 \\ &= \langle(x_{i1}, x_{i2}), (x_{j1}, x_{j2})\rangle^2 \end{aligned} \tag{2.3}$$

The result in Equation 2.3 shows that the inner product in $\mathcal{F}$ equals the inner product to the power of d in $\mathcal{X}$. This is attractive because whereas the computational effort in $\mathcal{F}$ scales with the number of dimensions (see Equation 2.2), in $\mathcal{X}$ it scales with the number of instances (Jäkel et al., 2007). In the case of a monomial feature map ϕ, it is not required to map the observations x_i and x_j to feature space to compute the inner product: it is sufficient to calculate the inner product in the input space and take it to the power of d (Jäkel et al., 2007). The combination of an inner product and ϕ defines a kernel (Equation 2.4), short for kernel function, k(x_i, x_j). The fact that kernels enable us to obtain the same superior classification performance as in feature space, for a much lower computational cost in input space, is called the kernel trick.

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{2.4}$$

The example above uses a polynomial kernel. The polynomial kernel including tuning parameters and two other widely used kernels are displayed in Table 2.1 (Shawe-Taylor and Cristianini, 2004; Park et al., 2012). The choice of the kernel function is largely dependent upon the data and can generally be determined by cross-validation (Fan, 2009).

Table 2.1: Some examples of kernels

Linear kernel	$k(x_i, x_j) = \langle x_i, x_j \rangle$
Gaussian kernel	$k(x_i, x_j) = exp(-\frac{\|x_i - x_j\|^2}{2\sigma})$
Polynomial kernel	$k(x_i, x_j) = (\gamma \langle x_i, x_j \rangle + r)^d$

d,r $\in \mathbb{N}$, $\gamma \in \mathbb{R}^+$

In the next section we discuss a combination of kernels and Random Forest: Kernel-Induced Random Forest (Fan, 2009).

2.3.3 Kernel-Induced Random Forest

Kernel-Induced Random Forest (KIRF) (Fan, 2009) differs from Random Forest in that, during the training phase, the former selects a random subgroup of observations and uses their kernel functions as splitting candidates, while the latter selects a random subgroup of variables as splitting candidates (Fan, 2009).

During the scoring phase, each splitting observation x_i (i=1,2,3,...,N) can be used with all observations that need to be scored x_k (k=1,2,3,...,M) in a kernel function $k(x_i, .)$ resulting in what can be conceived of as a novel feature $k_i(x_k) = k(x_i, x_k)$. For N splitting observations, there will be N such novel, kernel-induced features, $k_i(.)$, i=1,2,3,...,N (Fan, 2009). These novel, kernel-induced, features can then be employed by a binary recursive space partitioning tree in order to get pure divisions of the criterion variable in the terminal node.

Although KIRF has been shown to have excellent performance (Fan, 2009), the algorithm has the disadvantage that although the size of the kernel matrix K scales only linearly with the number of observations in the original data, K can quickly grow too large for Random Forest and subsequently KIRF. Kernel Factory alleviates this problem by randomly splitting the training data into a number of mutually exclusive partitions. This approach reduces the probability of surpassing limitations of software packages by several orders of magnitude (e.g., for R the maximum size of an object is $2^{31} - 1$ elements (R Core Team, 2012) and increases computational speed by several orders of magnitude (because kernel matrices can now be computed in parallel more easily).

2.4 Kernel Factory

The training phase (see Algorithm 1) starts by partitioning the data into mutually exclusive row and column partitions followed by scaling. Each partition forms an input space and is transformed by a kernel function into a kernel matrix K (categorical features are not used in the computation of K but are added afterwards). Subsequently, each K is used as training data for a binary base classifier (Random Forest).

The kernel function (polynomial, linear, or radial basis) can be (1) user specified, (2) randomly chosen per partition, or (3) determined by sequentially applying all kernels to the first partition, assessing predictive performance on a hold- out validation set, and selecting the best performing kernel.

This process results in a number of predictions equal to the number of partitions. We use a weighted average to combine the predictions into one final prediction in which the weights are optimized using a genetic algorithm (GA) (see Algorithm 2). For the fitness score we employ the predictive performance (area under the receiver operating characteristic curve) on a hold out sample. In extant literature, several studies have used a GA to weight the predictions of ensemble members before combination. To the best of our knowledge Yao and Liu (1998) were the first to use a GA in this context (for a neural network ensemble). Later, Zhou et al. (2002) showed that using the evolved weights to make up the neural network ensemble yields superior performance to normal averaging (see also Kim et al., 2002). Finally, Sylvester and Chawla (2005) use a GA to combine trees. All studies point to the beneficial effects of GAs in ensemble formation on predictive performance.

The prediction phase (see Algorithm 3) starts by constructing the same column partitions and applying scaling as in the training set. Next, the features from the training set are used to compute the Ks. Finally, the weighted average of all predictions is computed using the weights obtained from the genetic algorithm.

2.5 Empirical study

2.5.1 Data

Our experiments use 11 data sets from the UCI repository and 3 synthetic data sets. The first 11 data sets were selected because we have used them in past research, because the dependent variable is binary and because they contain continuous predictor variables (required for kernels).

```
Input:
• $cp$=Number of column partitions
• $rp$=Number of row partitions
• Method= polynomial kernel function (pol), linear kernel function (lin), radial basis kernel function (rbf), random choice (ran={pol, lin, rbf}), burn- in choice of best function (burn={pol, lin, rbf})
• Input space $\mathcal{X} = (x_1, \dots, x_n)$ with class labels ($Y$={0,1}) where $n$ is the number of features and let $N$ be the number of objects

Randomize order of rows and columns of $\mathcal{X}$
Divide $\mathcal{X}$: 80% of $r$ into $\mathcal{X}_{training}$ and 20% of $r$ into $\mathcal{X}_{validation}$
Scale both $\mathcal{X}_{training}$ and $\mathcal{X}_{validation}$: for every feature $\frac{\mathcal{X}_p}{range(\mathcal{X}_{p,training})}$
For $\mathcal{X}_{training}$: create $rp$ times $cp$ partitions of equal size ($p_{training}$)
For $\mathcal{X}_{validation}$: create $cp$ partitions of equal size ($p_{validation}$)
for every partition $p_{training}$ do
    • Apply method:
        ○ Let
            ■ lin=$\langle x_i, x_j \rangle$
            ■ rbf=$exp(-\frac{\|x_i - x_j\|^2}{2}$
            ■ pol=$\langle x_i, x_j \rangle^2$
        ○ If method==lin then select lin
        Else if method==pol then select pol
        Else if method==rbf then select rbf
        Else if method==ran then randomly select one of {pol, lin, rbf}
        Else if method==burn then
            ■ If $p_{training}$ ==1: for every method in {pol, lin, rbf} do
                • Compute kernel matrix $K_{training}$ on numeric features
                • Augment $K_{training}$ with raw features
                • Build classifier $h$ on $K_{training}$
                • Compute $K_{validation}$, populated by elements where, for the kernel function, observations $x_i$ come from $p_{validation}$ (with same numeric features as in $p_{training}$ ==1) and $x_j$ come from numeric features of $p_{training}$ ==1
                • Augment $K_{validation}$ with the raw features from $p_{validation}$
                • Deploy $h : \hat{Y}_{prob} = h(K_{validation})$
                • Compute AUC
                end
            ■ Select one of {pol, lin, rbf} which yielded max AUC
    • Compute kernel matrix $K_{training}$ on numeric features
    • Augment $K_{training}$ with raw features
    • Build classifier $h$ on $K_{training}$
    • Compute $K_{validation}$, populated by elements where, for the kernel function, observations $x_i$ come from $p_{validation}$ (with same numeric features as in current $p_{training}$) and $x_j$ come from numeric features of current $p_{training}$
    • Augment $K_{validation}$ with the raw features from $p_{validation}$
    • Deploy $h : \hat{Y}_{prob} = h(K_{validation})$
end
```

algorithm 1: Pseudo code for part 1 of the estimation phase of Kernel Factory

```
Optimize classifier weights for weighted averaging of predicted probabilities with genetic algorithm on validation space
    • Choose initial population of sets of weights
    • Repeat 100 times
        ◦ For each set of weights $\omega = (w_1, \dots, w_L)$ compute weighted average for a given object $x_i$ having a set of $\hat{Y}_{prob,i} = (\hat{Y}_{i,1}, \dots, \hat{Y}_{i,L})$ , with $L = (1, \dots, rp\ times\ cp)$ the number of classifiers (= number of weights):
              $g_i(\hat{Y}_{prob}) = \sum_{l=1}^{L} w_l(\hat{Y}_{prob,i,l})$
        ◦ Evaluate fitness (AUC)
        ◦ Select best-fit set of weights
        ◦ Apply crossover operator
        ◦ Apply mutation operator
    • Choose set of optimal weights (weights with best fit; highest AUC): $\omega = (w_1, \dots, w_L)$ with $L = rp\ times\ cp$
```

algorithm 2: Pseudo code for part 2 of the estimation phase of Kernel Factory

```
Input:
• Input space $\mathcal{X}_{new} = (x_1, \dots, x_n)$ where $n$ is the number of features and let $N$ be the number of objects

Apply order of rows and columns of $\mathcal{X}_{training}$ to $\mathcal{X}_{new}$
Scale $\mathcal{X}_{new}$: for every feature $\frac{\mathcal{X}_p}{range(\mathcal{X}_{p,training})}$
For $\mathcal{X}_{new}$: create $cp$ partitions ($p_{new}$) of equal size
for every classifier $h$ do
    • Compute $K_{new}$ on numeric features populated by elements where, for the kernel function, observations $x_i$ come from $p_{new}$ (with same numeric features as in current $p_{training}$) and $x_j$ come from numeric features of current $p_{training}$
    • Augment the kernel matrix with the raw features from $p_{new}$
    • Deploy $h$: $\hat{Y}_{prob} = h(K_{new})$
end
Compute weighted average using optimal weights: for a given object $x_i$ having a set of $\hat{Y}_{prob,i} = (\hat{Y}_{i,1}, \dots, \hat{Y}_{i,L})$, with $L = (1, \dots, rp\ times\ cp)$ the number of classifiers (= number of weights), use optimal $\omega$:
      $g_i(\hat{Y}_{prob}) = \sum_{l=1}^{L} w_l(\hat{Y}_{prob,i,l})$
$g_i$ is the confidence that object $x_i$ has $Y = 1$
```

algorithm 3: Pseudo code for the prediction phase of Kernel Factory

Moreover, most of them are used in Breiman's Random Forest paper (Breiman, 2001). The 3 synthetic data sets are constructed using the R package $mlbench$ (Leisch and Dimitriadou, 2012) and are chosen because KIRF improves upon Random Forest (Fan, 2009) on these data sets. The underlying reason is that, in these specific data sets, the radial basis function (RBF) helps Random Forest to classify the Gaussian distribution present in these data sets. It is well documented that using kernel functions only has a positive effect on predictive performance if the right kernel function is used and even has a harmful effect if the wrong one is used. Hence, KIRF should only be used with the right kernel and only if this improves upon Random Forest. Therefore, it makes little sense to compare Random Forest and KIRF blindly (without testing different kernels) on different data sets and make generalizations about which algorithm is best (Fan, 2009). Random Forest and KIRF (with different kernels) should always be tested together and the best algorithm should be selected. Hence, to give KIRF an honest chance in a comparison with the other algorithms, we use these synthetic data sets because we know they have Gaussian distributions and that a RBF kernel will add positively to the predictive performance.

In general, we expect that Kernel Factory will improve on KIRF because of the internal kernel selection procedure (in case of method=*burn* or *random*). In either case, Kernel Factory will not require manually testing multiple kernels.

Table 2.2 gives a brief summary of the data sets. N is the number of observations, n is the number of predictor variables

Table 2.2: Properties of the data sets used in the empirical study

Data	N	n, continuous	n categorical
Heart (Cleveland)	303	5	8
Hepatitis	155	6	13
Ionosphere	351	32	1
Pima (Diabetes)	768	8	0
Credit	690	6	9
Sonar	208	59	0
Wdbc (Cancer)	569	30	0
HeartHun (Hungary)	294	5	7
GermanCredit	1000	7	13
AustralianCredit	690	6	8
HorseColic	368	9	13
Ringnorm	1000	10	0
Peak	1000	6	0
Circle	1000	20	0

The last three data sets are the synthetic ones. The criterion variable was initially continuous in the Peak data set and is transformed to a binary variable by assigning a 1 if the value is greater than the mean and 0 otherwise.

2.5.2 Implementations of Algorithms

Random Forest requires two parameters to be set: the number of trees and the number of variables to try at each split. We follow Breiman's recommendation (Breiman, 2001) to use a large number of trees (1000) and the square root of the total number of variables as the number of predictors.

For KIRF we used the same settings as for Random Forest. For the Gaussian radial basis function we set the Gaussian kernel parameter σ equal to 1 because a decision tree is invariant to monotonic transformations of data (Fan, 2009). For the polynomial kernel function we used a degree of 2, a scale of 1 and an offset of 0. The categorical predictor variables are not involved in the computation of kernels but are kept as extra attributes during tree construction.

In using Kernel Factory, two parameters to set are the number of row and column partitions. Because the algorithm is designed to overcome practical limitations of computational resources, we chose parameter values aimed at accommodating the server we used for our experiments. We used $int(log_{10}(N+1))$ for the number of row partitions and both $ceil(log_5(n+1))$ and 1 for the number of column partitions. We also wanted to test with one column partition because column partitions are rather meant to introduce extra diversity in the ensemble members while it are the row partitions have a much bigger impact on the speed of the algorithm due to their direct impact on the size of the Ks. Setting the number of column partitions to 1 and comparing it to $ceil(log_5(n+1))$ allows us to understand the impact of that process. The base algorithm was Random Forest, using the same settings as above. The kernel parameter settings are also identical to the ones we used in KIRF. We employed a population size of 100, 200 iterations and a mutation chance of 0.01 for the Genetic Algorithm.

We have submitted an open-source R-software package of the algorithm ($kernelFactory$) to CRAN (Ballings and Van den Poel, 2013). Packages that are used by Kernel Factory, KIRF and Random Forest are $kernlab$ (Karatzoglou et al., 2004), $randomForest$ (Liaw and Wiener, 2002), $genalg$ (Willighagen, 2012), and $ROCR$ (Sing et al., 2009).

2.5.3 Model Performance Evaluation

To evaluate the performance of a model we use accuracy or percentage correctly classified (PCC) and, the area under the receiver operating characteristic curve (AUC or AUROC). PCC is defined as follows:

$$PCC = \frac{TP + TN}{P + N} \tag{2.5}$$

with TP: True Positives, TN: True Negatives, P: Positives (event), N: Negatives (non-event)

An important disadvantage of PCC is that it is sensitive to the chosen cut-off value of the posterior probabilities (Baecke and Van den Poel, 2012; Thorleuchter and Van den Poel, 2012) that decides when an object is predicted to be of class zero or one. While accuracy is the performance of a model at only one cut-off value AUC is the performance of a model across all threshold values. Several authors (Provost et al., 1998; Langley, 2000; Coussement and Van den Poel, 2008) argue AUC to be an objective criterion for classifier performance. As such AUC is a more adequate performance measure than PCC (Baecke and Van den Poel, 2011).

More formally, the receiver operating characteristic (ROC) curve is obtained from plotting sensitivity and 1-specificity considering all possible cut-off values (Hanley and Mcneil, 1982). AUC ranges from .5, if the predictions are not better than random, to 1, if the model predicts the event perfectly (Baecke and Van den Poel, 2011). AUC is defined as follows:

$$AUC = \int_0^1 \frac{TP}{(TP+FN)} d\frac{FP}{(FP+TN)} \tag{2.6}$$

,with TP: True Positives, FN: False Negatives, FP: False Positives, TN: True Negatives, P: Positives (event), N: Negatives (non-event)

Reported performance measures are all medians over five times two-fold cross- validation (5x2cv) (Dietterich, 1998; Alpaydin, 1999). This cross- validation procedure randomly divides the sample in two equal parts and repeats this process five times. Each part is used both as a training and validation part. This results in ten performance measures per model (Dietterich, 1998). The same splits are used for all algorithms. We also report the inter quartile range as a measure of dispersion.

In order to determine whether models are significantly different in terms of AUC or PCC, we follow Demsar's recommendation (Demsar, 2006) to use the nonparametric Friedman test with Nemenyi's post-hoc test (Nemenyi, 1963) for comparisons of the algorithms. In this context we report the algorithms' average ranks per data set. Algorithms are ranked, per fold separately, with the best algorithm receiving rank 1, the second receiving rank of 2, an so forth. It is important to note that this approach incorporates the relatedness of the folds (algorithm ranks are computed per fold and then the average rank is computed per data set). The folds are not treated as independent as is the case when computing the median. In other words, when ranks are computed the order of the folds is preserved and comparisons are made per fold, whereas when the median is computed the order of the folds is not preserved because they are sorted by predictive performance and the middle one is selected. Hence, computing ranks in this fashion allows stricter comparison than computing the median.

We opted for testing on the folds per data set, as opposed to testing the median across the data sets, because the data sets' predictive performances are not commensurate. Hence, controlling for the family-wise error, the probability of at least one false positive in any of the comparisons, is debatable because the costs of these false positives differs across data sets (also see Webb, 2000): the data sets are not a family. Moreover, there are no tests available for multiple data sets that can consider the folds for each data set (Demsar, 2006). More concretely, although we will compare KIRF and Kernel Factory on all data sets it makes no sense to blindly compare Random Forest with KIRF and Kernel Factory on all data sets. As aforementioned, the choice of the kernel depends on the data and can be determined by cross-validation (Fan, 2009). Hence, it is obvious that KIRF nor Kernel Factory will be used if they perform considerably worse than Random Forest due to the wrong kernel choice. In sum, because we want to make declarations per data set, and not merely per algorithm, we opted to test on the folds.

2.5.4 Results

Table 2.3 and Table 2.5 show the median PCC and AUC respectively of the 10 cross-validation folds. In all tables and the rest of this text, KF stands for Kernel Factory, KIRF stands for Kernel-Induced Random Forest, RF stands for Random Forest and $cp1$ stands for column partitions equal to one. If $cp1$ is not specified along RF it means that $ceil(log5(n+1))$ is used to determine the number of column partitions. Table 2.4 and 2.6 contain the inter quartile ranges for PCC and AUC respectively. Table 2.7 (PCC) and Table 2.9 (AUC) report the average ranks (lower is better) and Table 2.8 (PCC) and Table 2.10 (AUC) report selected differences of the average rankings. As mentioned in the model performance evaluation section, taking the median of the folds does not respect that performances are related per fold, whereas taking the Friedman test does (Demsar, 2006). This is also the reason why results can sometimes deviate to a limited extent. It has to be noted that using the rank as opposed to the median can be considered a stricter comparison of predictive performance.

The performance of two classifiers differs significantly if their average ranks differ (see Table 2.8 and 2.10) by at least the critical difference of 6.2748 (which is the critical difference for a p-value of 0.05, 14 classifiers and 10 folds) (see Demsar (2006) for the formula). The significant differences in Table 2.8 and Table 2.10 are put in bold.

Comparing (1) KFrbf and KFrbfcp1, (2) KFpol and KFpolcp1, (3) KFlin and KFlincp1, (4) KFran in KFrancp1, and (5) KFburn and KFburncp1 in both Table 2.3 and Table 2.5 we observe that increasing the number of column partitions, as such introducing more diversity, does not have a consistent effect across the data sets. Even more so, on the data sets that we created with a Gaussian distribution (the bottom three data sets in Table 2.3), setting the number of column partitions to 1 has a beneficial effect for the KFs with a radial basis function (KFrbfcp1, KFrancp1, KFburncp1). Most of these benefits are also significant (see column KFrbf-KFrbfcp1, KFran-KFrancp1, KFburn-KFburncp1 in Table 2.8 and Table 2.10). Note that method *ran* will have a probability of one third to select RBF per partition and method *burn* will select the best performing kernel function on the first partition and use that kernel for all partitions. In this case, for the last three data sets, it was always the radial basis function that has been selected by KFburncp1. In Table 2.3, there are very low PCCs for KFpol, KFlin, KFran, and KFburn for the German Credit data set. In Table 2.5 the same can be observed for KFpolcp1 and KFlincp1 for the Circle data set. Moreover Table 2.4 and Table 2.6 also show a high inter quartile range (IQR) for these algorithms on these data sets. These are strong signs of overfitting and is likely to be caused by the high dimensionality of the Ks.

Table 2.3: The median of the 10 folds for PCC

PCC Me-dian	KF rbf	KF rbf cp1	KIRF rbf	KF pol	KF pol cp1	KIRF pol	KF lin	KF lin cp1	KIRF lin	KF ran	KF ran cp1	KF burn	KF burn cp1	RF
Hea	0.7591	0.7228	0.6755	0.7285	0.7492	0.6457	0.7318	0.7185	0.6601	0.7252	0.7360	0.7426	0.7393	0.8212
Hep	0.8129	0.8194	0.8141	0.8064	0.8259	0.8258	0.8129	0.8000	0.8193	0.8193	0.8129	0.7821	0.8129	0.8442
Ion	0.9003	0.9060	0.9259	0.8807	0.9145	0.9371	0.8889	0.9088	0.9288	0.9059	0.9205	0.8892	0.9062	0.9345
Pim	0.7174	0.7396	0.7435	0.7135	0.7448	0.7357	0.7109	0.7370	0.7344	0.7292	0.7344	0.7292	0.7383	0.7565
Cre	0.7913	0.8174	0.7377	0.7957	0.8130	0.7000	0.8217	0.8014	0.7014	0.7870	0.7957	0.8058	0.7971	0.8725
Son	0.7452	0.7500	0.7933	0.7308	0.7548	0.7067	0.7308	0.7500	0.7115	0.7356	0.7404	0.7356	0.7500	0.7981
wdb	0.9419	0.9472	0.9439	0.9455	0.9421	0.9437	0.9437	0.9473	0.9419	0.9438	0.9420	0.9561	0.9438	0.9542
Hea	0.7551	0.7823	0.7449	0.7619	0.7517	0.7041	0.7721	0.7517	0.7007	0.7483	0.7585	0.7279	0.7687	0.8095
Ger	0.6900	0.7050	0.6700	0.4910	0.7040	0.6660	0.3050	0.7130	0.6640	0.3050	0.7040	0.3040	0.7050	0.7550
Aus	0.7855	0.7986	0.7391	0.7971	0.8000	0.6942	0.8000	0.7913	0.7014	0.7913	0.8087	0.8000	0.7942	0.8638
Hor	0.7147	0.7255	0.7201	0.7147	0.7092	0.7011	0.7255	0.7120	0.6984	0.6902	0.7337	0.7228	0.7201	0.7745
Rin	0.8850	0.9290	0.9270	0.8720	0.8960	0.9020	0.8780	0.8950	0.9020	0.8840	0.8930	0.8800	0.9290	0.8910
Pea	0.9100	0.9840	0.9890	0.8930	0.9420	0.9400	0.8860	0.9340	0.9410	0.8920	0.9830	0.8960	0.9870	0.9270
Cir	0.8330	0.8810	0.8990	0.8330	0.8330	0.8330	0.8330	0.8320	0.8330	0.8330	0.8520	0.8330	0.8760	0.8350

Table 2.4: The inter quartile range of the 10 folds for PCC

PCC IQR	KF rbf	KF rbf cp1	KIRF rbf	KF pol	KF pol cp1	KIRF pol	KF lin	KF lin cp1	KIRF lin	KF ran	KF ran cp1	KF burn	KF burn cp1	RF
Hea	0.0342	0.0498	0.0198	0.0517	0.0362	0.0521	0.0383	0.0809	0.0560	0.0553	0.0607	0.0756	0.0514	0.0243
Hep	0.0399	0.0536	0.0617	0.0303	0.0611	0.0597	0.0512	0.0718	0.0486	0.0373	0.0304	0.0364	0.0503	0.0662
Ion	0.0409	0.0357	0.0243	0.0406	0.0280	0.0255	0.0287	0.0280	0.0313	0.0595	0.0372	0.0455	0.0346	0.0270
Pim	0.0234	0.0286	0.0208	0.0273	0.0124	0.0286	0.0410	0.0280	0.0312	0.0202	0.0228	0.0267	0.0345	0.0234
Cre	0.0384	0.0428	0.0341	0.0123	0.0319	0.0609	0.0565	0.0232	0.0536	0.0355	0.0232	0.0507	0.0181	0.0232
Son	0.0529	0.0529	0.0745	0.0601	0.0457	0.0529	0.0409	0.0505	0.0721	0.0361	0.0264	0.0601	0.0745	0.0625
wdb	0.0141	0.0204	0.0086	0.0149	0.0183	0.0168	0.0133	0.0132	0.0159	0.0158	0.0262	0.0132	0.0070	0.0096
Hea	0.0833	0.0561	0.0374	0.0833	0.0255	0.0680	0.0816	0.0731	0.0646	0.0476	0.0629	0.0697	0.0374	0.0272
Ger	0.3995	0.0170	0.0250	0.3965	0.0210	0.0155	0.2890	0.0255	0.0200	0.0160	0.0125	0.0185	0.0220	0.0095
Aus	0.0159	0.0536	0.0536	0.0529	0.0138	0.0399	0.0601	0.0341	0.0312	0.0254	0.0500	0.0377	0.0486	0.0109
Hor	0.0204	0.0462	0.0312	0.0312	0.0380	0.0190	0.0177	0.0217	0.0217	0.0557	0.0435	0.0326	0.0353	0.0190
Rin	0.0140	0.0070	0.0110	0.0130	0.0170	0.0065	0.0125	0.0135	0.0065	0.0115	0.0195	0.0120	0.0055	0.0280
Pea	0.0155	0.0090	0.0075	0.0150	0.0245	0.0140	0.0205	0.0180	0.0150	0.0190	0.0330	0.0195	0.0105	0.0235
Cir	0.0150	0.0270	0.0245	0.0150	0.0150	0.0150	0.0150	0.0140	0.0150	0.0150	0.0185	0.0150	0.0150	0.0150

Table 2.5: The median of the 10 folds for AUC

AUC Me-dian	KF rbf	KF rbf cp1	KIRF rbf	KF pol	KF pol cp1	KIRF pol	KF lin	KF lin cp1	KIRF lin	KF ran	KF ran cp1	KF burn	KF burn cp1	RF
Hea	0.8286	0.8081	0.7341	0.8267	0.8345	0.6805	0.7973	0.8167	0.6800	0.7947	0.8251	0.8407	0.8226	0.8993
Hep	0.8276	0.8199	0.8188	0.7918	0.7864	0.7915	0.8006	0.8095	0.7978	0.7961	0.8227	0.7927	0.8213	0.8681
Ion	0.9478	0.9528	0.9701	0.9583	0.9558	0.9746	0.9644	0.9589	0.9747	0.9559	0.9633	0.9583	0.9563	0.9756
Pim	0.7712	0.8046	0.8047	0.7795	0.8007	0.7978	0.7851	0.7952	0.8004	0.7855	0.7995	0.7919	0.7996	0.8188
Cre	0.8768	0.8863	0.8003	0.8714	0.8849	0.7639	0.8767	0.8852	0.7656	0.8760	0.8748	0.8709	0.8781	0.9301
Son	0.8609	0.8617	0.8865	0.8481	0.8519	0.8212	0.8456	0.8490	0.8219	0.8454	0.8514	0.8500	0.8586	0.9156
wdb	0.9886	0.9888	0.9874	0.9872	0.9857	0.9844	0.9880	0.9872	0.9850	0.9853	0.9879	0.9864	0.9853	0.9904
Hea	0.8369	0.8276	0.7969	0.8349	0.8202	0.7305	0.8144	0.8285	0.7331	0.8178	0.8215	0.8127	0.8275	0.8894
Ger	0.6737	0.6564	0.5976	0.6611	0.6701	0.5817	0.6535	0.6570	0.5838	0.6791	0.6610	0.6677	0.6801	0.7881
Aus	0.8768	0.8769	0.8074	0.8818	0.8752	0.7591	0.8730	0.8748	0.7568	0.8802	0.8867	0.8863	0.8756	0.9337
Hor	0.7889	0.7613	0.7621	0.7763	0.7562	0.7262	0.7763	0.7606	0.7247	0.7764	0.7800	0.7653	0.7669	0.8353
Rin	0.9554	0.9794	0.9788	0.9505	0.9617	0.9653	0.9527	0.9627	0.9654	0.9581	0.9610	0.9565	0.9796	0.9555
Pea	0.9749	0.9995	0.9996	0.9620	0.9927	0.9940	0.9600	0.9927	0.9944	0.9675	0.9991	0.9676	0.9996	0.9868
Cir	0.8915	0.9864	0.9867	0.6212	0.4286	0.5245	0.6078	0.4245	0.5239	0.7283	0.7352	0.8789	0.9834	0.7748

Table 2.6: The inter quartile range of the 10 folds for AUC

AUC IQR	KF rbf	KF rbf cp1	KIRF rbf	KF pol	KF pol cp1	KIRF pol	KF lin	KF lin cp1	KIRF lin	KF ran	KF ran cp1	KF burn	KF burn cp1	RF
Hea	0.0379	0.0639	0.0231	0.0365	0.0334	0.0734	0.0547	0.0737	0.0587	0.0614	0.0999	0.0944	0.0556	0.0187
Hep	0.0532	0.0407	0.0897	0.0474	0.0762	0.1133	0.0860	0.0396	0.1210	0.1004	0.0908	0.1045	0.0521	0.0664
Ion	0.0156	0.0261	0.0207	0.0236	0.0196	0.0191	0.0174	0.0213	0.0201	0.0095	0.0198	0.0058	0.0160	0.0113
Pim	0.0342	0.0358	0.0225	0.0524	0.0178	0.0207	0.0264	0.0340	0.0238	0.0191	0.0171	0.0195	0.0382	0.0155
Cre	0.0331	0.0385	0.0435	0.0437	0.0227	0.0616	0.0328	0.0272	0.0550	0.0462	0.0224	0.0376	0.0268	0.0221
Son	0.0442	0.0351	0.0314	0.0242	0.0578	0.0317	0.0231	0.0348	0.0300	0.0297	0.0255	0.0238	0.0413	0.0320
wdb	0.0065	0.0073	0.0108	0.0031	0.0091	0.0070	0.0055	0.0060	0.0073	0.0053	0.0065	0.0085	0.0076	0.0052
Hea	0.0328	0.0257	0.0612	0.0318	0.0330	0.0753	0.0905	0.0633	0.0753	0.0700	0.0348	0.0600	0.0326	0.0362
Ger	0.0466	0.0348	0.0302	0.0548	0.0189	0.0253	0.0471	0.0094	0.0256	0.0365	0.0249	0.0441	0.0398	0.0104
Aus	0.0346	0.0492	0.0367	0.0266	0.0093	0.0483	0.0291	0.0284	0.0464	0.0188	0.0377	0.0287	0.0318	0.0177
Hor	0.0325	0.0303	0.0313	0.0408	0.0381	0.0465	0.0189	0.0471	0.0483	0.0661	0.0610	0.0438	0.0401	0.0208
Rin	0.0093	0.0029	0.0016	0.0093	0.0064	0.0047	0.0066	0.0045	0.0054	0.0049	0.0113	0.0032	0.0025	0.0064
Pea	0.0038	0.0005	0.0004	0.0077	0.0030	0.0026	0.0070	0.0029	0.0025	0.0041	0.0033	0.0068	0.0006	0.0060
Cir	0.0298	0.0080	0.0039	0.0986	0.0553	0.0576	0.0627	0.1275	0.0611	0.0844	0.5141	0.0403	0.0085	0.0499

Table 2.7: Average rankings of the folds (per data set) for PCC

Avg. rank-ings (PCC)	KF rbf	KF rbf cp1	KIRF rbf	KF pol	KF pol cp1	KIRF pol	KF lin	KF lin cp1	KIRF lin	KF ran	KF ran cp1	KF burn	KF burn cp1	RF	Friedman $\chi^2(13)$, p<
Hea	5.00	6.75	11.75	6.25	5.70	13.55	8.30	6.55	12.55	6.45	7.65	6.55	6.65	1.30	77.98, .001
Hep	7.85	8.15	8.15	8.70	5.35	5.95	9.00	7.70	5.95	8.85	7.85	10.35	8.85	2.30	32.16, .001
Ion	9.25	7.60	5.15	10.50	7.60	3.40	10.40	8.35	4.00	9.00	6.25	11.75	9.00	2.75	59.66, .001
Pim	11.05	5.55	6.25	11.10	5.70	5.65	10.05	7.15	6.90	9.20	6.40	9.40	7.25	3.35	40.25, .001
Cre	7.50	5.70	11.60	7.30	4.75	13.45	6.00	5.95	13.15	8.00	7.45	6.10	7.05	1.00	82.84, .001
Son	8.25	6.20	3.70	7.70	7.65	11.85	9.50	6.65	10.20	7.80	8.35	8.20	6.90	2.05	46.19, .001
wdb	8.70	6.55	6.70	6.15	9.15	8.50	6.85	6.85	8.05	8.55	10.35	5.30	8.90	4.40	20.72, .005
Hea	7.15	6.15	8.20	8.00	7.25	11.70	8.10	6.50	11.70	7.80	6.60	8.10	5.35	2.40	41.33, .001
Ger	8.75	5.40	9.05	9.05	4.50	9.75	10.80	4.50	10.05	11.65	4.70	11.00	4.80	1.00	80.07, .001
Aus	7.40	6.80	11.45	5.80	6.20	13.50	6.05	7.30	13.45	6.80	5.80	7.30	6.05	1.10	80.71, .001
Hor	8.20	7.05	6.35	7.75	8.15	10.20	7.10	7.25	10.85	9.60	5.65	8.10	7.50	1.25	39.58, .001
Rin	9.70	2.00	2.15	11.90	7.25	5.60	11.55	7.65	6.25	10.45	7.20	11.25	2.05	10.00	93.88, .001
Pea	10.35	2.55	1.60	12.35	6.90	6.30	13.00	6.60	6.30	12.30	4.25	11.65	2.35	8.50	117.19, .001
Cir	9.20	2.75	1.25	9.65	9.65	9.65	9.65	10.05	9.65	9.15	6.20	9.15	2.55	6.45	100.72, .001

Table 2.8: Selected differences of the average rankings of the folds (per data set) for PCC

Selected differences of average rankings	KF rbf- KF rbf cp1	KIRF rbf - KF rbf cp1	KIRF rbf - KF rbf	KF pol - KF pol cp1	KIRF pol - KF pol cp1	KIRF pol - KF pol	KF lin - KF lin cp1	KIRF lin - KF lin cp1	KIRF lin - KF lin	KF ran - KF ran cp1	KF burn - KF burn cp1	KF ran - KF burn	KF ran cp1 - KF burn cp1	RF - KIRF rbf	RF - KF rbf cp1
Hea	-1.75	5.00	**6.75**	0.55	**7.85**	**7.30**	1.75	6.00	4.25	-1.20	-0.10	-0.10	1.00	**-10.45**	-5.45
Hep	-0.30	0.00	0.30	3.35	0.60	-2.75	1.30	-1.75	-3.05	1.00	1.50	-1.50	-1.00	-5.85	-5.85
Ion	1.65	-2.45	-4.10	2.90	-4.20	**-7.10**	2.05	-4.35	**-6.40**	2.75	2.75	-2.75	-2.75	-2.40	-4.85
Pim	5.50	0.70	-4.80	5.40	-0.05	-5.45	2.90	-0.25	-3.15	2.80	2.15	-0.20	-0.85	-2.90	-2.20
Cre	1.80	5.90	4.10	2.55	**8.70**	6.15	0.05	**7.20**	**7.15**	0.55	-0.95	1.90	0.40	**-10.60**	-4.70
Son	2.05	-2.50	-4.55	0.05	4.20	4.15	2.85	3.55	0.70	-0.55	1.30	-0.40	1.45	-1.65	-4.15
wdb	2.15	0.15	-2.00	-3.00	-0.65	2.35	0.00	1.20	1.20	-1.80	-3.60	3.25	1.45	-2.30	-2.15
Hea	1.00	2.05	1.05	0.75	4.45	3.70	1.60	5.20	3.60	1.20	2.75	-0.30	1.25	-5.80	-3.75
Ger	3.35	3.65	0.30	4.55	5.25	0.70	**6.30**	5.55	-0.75	**6.95**	6.20	0.65	-0.10	**-8.05**	-4.40
Aus	0.60	4.65	4.05	-0.40	**7.30**	7.70	-1.25	6.15	**7.40**	1.00	1.25	-0.50	-0.25	**-10.35**	-5.70
Hor	1.15	-0.70	-1.85	-0.40	2.05	2.45	-0.15	3.60	3.75	3.95	0.60	1.50	-1.85	-5.10	-5.80
Rin	**7.70**	0.15	**-7.55**	4.65	-1.65	**-6.30**	3.90	-1.40	-5.30	3.25	**9.20**	-0.80	5.15	**7.85**	**8.00**
Pea	**7.80**	-0.95	**-8.75**	5.45	-0.60	-6.05	**6.40**	-0.30	**-6.70**	**8.05**	**9.30**	0.65	1.90	**6.90**	5.95
Cir	**6.45**	-1.50	**-7.95**	0.00	0.00	0.00	-0.40	-0.40	0.00	2.95	**6.60**	0.00	3.65	5.20	3.70

Table 2.9: Average rankings of the folds (per data set) for AUC

Avg. rank-ings (AUC)	KF rbf	KF rbf cp1	KIRF rbf	KF pol	KF pol cp1	KIRF pol	KF lin	KF lin cp1	KIRF lin	KF ran	KF ran cp1	KF burn	KF burn cp1	RF	Friedman $\chi^2(13)$, p<
Hea	6.10	7.10	12.00	5.40	5.90	13.10	8.30	6.00	13.50	7.40	6.80	6.20	6.20	1.00	83.27, .001
Hep	4.90	6.80	7.60	8.40	9.70	9.85	7.20	7.90	9.35	7.20	6.70	10.50	6.50	2.40	33.63, .01
Ion	10.45	10.25	3.40	10.30	9.50	3.20	6.95	8.90	3.00	9.90	7.65	9.20	10.10	2.20	73.99, .001
Pim	10.50	5.80	7.70	8.80	5.70	7.90	10.40	6.30	8.00	10.30	6.80	9.50	6.00	1.30	45.79, .001
Cre	6.80	5.60	12.00	8.20	4.40	13.60	5.90	5.40	13.40	7.60	6.70	7.20	7.20	1.00	89.44, .001
Son	6.70	6.20	3.70	7.55	8.75	12.40	8.30	7.50	12.40	9.30	7.70	6.70	6.50	1.30	63.09, .001
wdb	7.35	4.30	6.70	7.20	7.75	10.60	7.00	7.80	9.45	10.15	8.70	6.60	7.40	4.00	26.49, 0.05
Hea	5.90	6.00	10.40	5.80	7.20	12.90	8.60	6.90	12.50	6.70	6.80	8.10	6.10	1.10	66.48, .001
Ger	5.70	7.80	11.80	7.70	6.20	13.20	6.90	7.10	13.20	5.20	7.00	6.30	5.90	1.00	80.48, .001
Aus	6.80	6.50	12.20	6.50	6.80	13.30	7.30	7.60	13.50	6.10	5.30	5.30	6.80	1.00	85.22, .001
Hor	5.50	8.60	7.90	6.90	9.00	11.30	5.85	7.10	12.00	8.30	5.50	7.85	8.20	1.00	53.19, .001
Rin	9.00	1.80	2.10	11.70	8.20	5.90	11.80	7.80	5.50	9.60	7.60	10.40	2.10	11.50	94.38, .001
Pea	10.40	2.55	1.75	12.70	7.30	5.80	13.20	6.90	5.70	11.80	3.80	11.90	2.30	8.90	121.52, .001
Cir	4.90	1.90	1.50	8.80	13.30	11.30	9.20	12.60	11.10	7.50	8.40	5.10	2.70	6.70	112, .001

Table 2.10: *Selected differences of the average rankings of the folds (per data set) for AUC*

Selected differences of average rankings	KF rbf- KF rbf cp1	KIRF rbf - KF rbf cp1	KIRF rbf - KF rbf	KF pol - KF pol cp1	KIRF pol - KF pol cp1	KIRF pol - KF pol	KF lin - KF lin cp1	KIRF lin - KF lin cp1	KIRF lin - KF lin	KF ran - KF ran cp1	KF burn - KF burn cp1	KF ran - KF burn	KF ran cp1 - KF burn cp1	RF - KIRF rbf	RF - KF rbf cp1
Hea	-1.00	4.90	5.90	-0.50	**7.20**	**7.70**	2.30	**7.50**	5.20	0.60	0.00	1.20	0.60	**-11.00**	-6.10
Hep	-1.90	0.80	2.70	-1.30	0.15	1.45	-0.70	1.45	2.15	0.50	4.00	-3.30	0.20	-5.20	-4.40
Ion	0.20	**-6.85**	**-7.05**	0.80	**-6.30**	**-7.10**	-1.95	-5.90	-3.95	2.25	-0.90	0.70	-2.45	**-1.20**	**-8.05**
Pim	4.70	1.90	-2.80	3.10	2.20	-0.90	4.10	1.70	-2.40	3.50	3.50	0.80	0.80	**-6.40**	-4.50
Cre	1.20	**6.40**	5.20	3.80	**9.20**	5.40	0.50	**8.00**	**7.50**	0.90	0.00	0.40	-0.50	-11.00	-4.60
Son	0.50	-2.50	-3.00	-1.20	3.65	4.85	0.80	4.90	4.10	1.60	0.20	2.60	1.20	-2.40	-4.90
wdb	3.05	2.40	-0.65	-0.55	2.85	3.40	-0.80	1.65	2.45	1.45	-0.80	3.55	1.30	-2.70	-0.30
Hea	-0.10	4.40	4.50	-1.40	5.70	**7.10**	1.70	5.60	3.90	-0.10	2.00	-1.40	0.70	**-9.30**	-4.90
Ger	-2.10	4.00	6.10	1.50	**7.00**	5.50	-0.20	6.10	**6.30**	-1.80	0.40	-1.10	1.10	**-10.80**	**-6.80**
Aus	0.30	5.70	5.40	-0.30	**6.50**	**6.80**	-0.30	5.90	6.20	0.80	-1.50	0.80	-1.50	**-11.20**	-5.50
Hor	-3.10	-0.70	2.40	-2.10	2.30	4.40	-1.25	4.90	6.15	2.80	-0.35	0.45	-2.70	**-6.90**	**-7.60**
Rin	**7.20**	0.30	**-6.90**	3.50	-2.30	-5.80	4.00	-2.30	**-6.30**	2.00	**8.30**	-0.80	5.50	**9.40**	**9.70**
Pea	**7.85**	-0.80	**-8.65**	5.40	-1.50	**-6.90**	**6.30**	-1.20	**-7.50**	**8.00**	**9.60**	-0.10	1.50	**7.15**	**6.35**
Cir	3.00	-0.40	-3.40	-4.50	-2.00	2.50	-3.40	-1.50	1.90	-0.90	2.40	2.40	5.70	5.20	4.80

Comparing (1) KFrbfcp1 and KIRFrbf, (2) KFpolcp1 and KIRFpol, and (3) KFlincp1 and KIRFlin shows that KF performs similarly or considerably higher, and rarely considerably lower. For example, in Table 2.3 and 2.5 on the Credit and Australian Credit data sets KF performs around .10 PCC and .10 AUC better than KIRF. This conclusion is confirmed when looking at the columns KIRFrbf-KFrbfcp1, KIRFpol-KFpolcp1, KIRFlin-KFlincp1 in Table 2.8 and Table 2.10. In Table 2.8 (PCC), all 4 significant differences are in favor of KF. In Table 2.10 (AUC), out of 9 significant differences, 7 are in favor of KF.

KF with method *ran* and *burn* are both designed to choose the kernel for the user. Hence a direct comparison is in order. Interestingly, from Table 2.3 and Table 2.5 we see that KFrancp1 and KFburncp1 are very competitive. None of the differences are significant in Table 2.8 and 2.10 (see columns KFran-KFburn and KFrancp1-KFburncp1).

Finally, a comparison of RF with KF and KIRF fulfills our expectations in that RF is superior to the other two when the wrong kernel is selected, or a kernel is selected when none is needed. In contrast, when the right kernel is known (or determined in advance by cross-validation), KIRF and KF do perform considerably better (see columns KFrbfcp1, KIRFrbf. KFrancp1, KFburncp1, and RF for the last three data sets in Table 2.3 and 2.5). On the 3 relevant data sets, KIRFrbf and KFrbfcp1 perform better than RF (significantly in most cases) (see columns RF-KIRFrbf and RF-KFrbfcp1 in Table 2.8 and 2.10).

2.6 Conclusions

In this study we propose an ensemble method for kernel machines. The training data is randomly split into a number of mutually exclusive partitions defined by a row and column parameter. Each partition forms an input space and is transformed by a kernel function into a kernel matrix K. Subsequently, each K is used as training data for a binary base classifier (Random Forest). This results in a number of predictions equal to the number of partitions. A weighted average combines the predictions into one final prediction. To optimize the weights, a Genetic Algorithm is used.

This approach has the advantage of simultaneously promoting (1) diversity, (2) accuracy, and (3) computational speed. (1) Diversity is fostered because the individual Ks are based on a subset of features and observations, (2) accuracy is sought using strong base classifiers, and (3) computational speed is obtained because the computation of each K can be parallelized.

Using five times two-fold cross- validation we benchmark the classification performance of Kernel Factory against Random Forest and Kernel-Induced Random Forest (KIRF). Our findings are fourfold.

First, the number of column partitions matters. While partitioning the columns is primarily a way of introducing diversity in the ensemble, we found that one partition works better than many. We recommend using one partition and creating more in case of numerical problems when computing the Ks (which is often the case in data sets with many features).

Second, Kernel Factory is significantly better than Kernel-Induced Random Forest (KIRF) on several data sets (and performs rarely significantly worse than KIRF). Along with the superior speed of Kernel Factory on large data sets, we recommend it over KIRF.

Third, the two methods (*random* and *burn-in*) that automatically select the kernel function perform equally well and using them is a viable strategy when the right kernel function is unknown in advance.

Fourth and final, when using a kernel is appropriate, and the right kernel is specified, both Kernel Factory and Kernel-Induced Random Forest outperform Random Forest significantly.

The main practical implication of this study is that problems involving large data sets, that otherwise would be impossible to analyze using KIRF, can now be analyzed using Kernel Factory. Hence, Kernel Factory opens a doorway to increases in classification performance by kernel functions. We have made available an open-source R-software package of the algorithm (kernelFactory) at CRAN (Ballings and Van den Poel, 2013).

2.7 Future Research and Limitations

We have six avenues for future research, which at the same time can be considered the limitations of this study. The first avenue is to try other base classifiers. In this study we use Random Forest as a base classifier because it is good at handling a large number of predictors, and because we wanted to make a direct comparison with KIRF. Other options would be Logistic Regression with variable selection techniques, or Support Vector Machines. Random Mulitnomial Logit (Prinzie and Van den Poel, 2008) is an example of an ensemble method that might benefit from kernels.

The second direction for future research is to use other kernel functions (e.g., Üstün et al., 2006). Particularly interesting developments are kernels for categorical data (see Li and Racine, 2007). As in KIRF, we exclude the categorical variables when computing the kernel matrix and add them afterwards. It might prove valuable to use kernel functions that can handle categorical data.

While parameter values of Kernel Factory are inspired by practical reasons (computational speed), a third direction is to optimize these parameters. More specifically, there is quite a large body of research investigating what the optimal values are for selection, mutation, and crossover in genetic algorithms (e.g., Dejong and Spears, 1991). It may prove valuable to integrate this research and investigate whether it applies to weight optimization for classifier ensembles. Moreover, while in this study we determine the number of partitions by taking the logarithm of the number of rows and columns, these parameters will probably benefit from optimization too. Although we compare different settings (two values for number of column partitions), future research should study this more in depth.

A fourth avenue is to investigate larger data sets. The true value of Kernel Factory over KIRF lies in large data. Hence, future research should take a special interest in data with a high number of observations and predictors.

The fifth direction for future research is ensemble pruning. Currently all ensemble members are used and weighted in the prediction phase. In order to make Kernel Factory less memory demanding and less computationally expensive members with weight close to zero could be excluded from the scoring phase. Zhou et al. (2002) demonstrate that this principle works quite well.

The sixth and final direction is to further explore the mechanism of Kernel Factory with a bias-variance decomposition of the classification error (e.g., Zhou et al., 2002). This will provide insight into to which factors Kernel Factory owes its strengths.

2.8 Acknowledgment

Both authors acknowledge the IAP research network grant No. P6/03 of the Belgian government (Belgian Science Policy).

2.9 References

Alpaydin, E., 1999. Combined 5 x 2 cv f test for comparing supervised classification learning algorithms. Neural Computation 11 (8), 1885–1892.

Baecke, P., Van den Poel, D., 2011. Data augmentation by predicting spending pleasure using commercially available external data. Journal of Intelligent Information Systems 36 (3), 367–383.

Baecke, P., Van den Poel, D., Nov. 2012. Including spatial interdependence in customer acquisition models: A cross-category comparison. Expert Systems with Applications 39 (15), 12105–12113.

Ballings, M., Van den Poel, D., 2013. R package kernelFactory: an ensemble of kernel machines.

Breiman, L., Oct. 2001. Random forests. Machine Learning 45 (1), 5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., 1984. Classification and regression trees. Wadsworth, Belmont, C A.

Buckinx, W., Van den Poel, D., 2005. Customer base analysis: partial defection of behaviourally loyal clients in a non-contractual FMCG retail setting. European Journal of Operational Research 164 (1), 252–268.

Coussement, K., Van den Poel, D., 2008. Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. Expert Systems with Applications 34 (1), 313–327.

Dejong, K., Spears, W., 1991. An Analysis of the Interacting Roles of Population-Size and Crossover in Genetic Algorithms. Vol. 496. Springer-Verlag Berlin, Berlin.

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

Dietterich, T. G., 1998. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10 (7), 1895–1923.

Duda, R., Hart, P., Stork, D., 2001. Nonmetric methods. In: Pattern Classification. Wiley, NY, pp. 394–452.

Fan, G., 2009. Kernel-induced classification tree and random forest. Technical report, Dept. of Statistics and Actuarial Science, University of Waterloo.

Hanley, J., Mcneil, B., 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology 143 (1), 29–36.

Jäkel, F., Schölkopf, B., Wichmann, F. A., Dec. 2007. A tutorial on kernel methods for categorization. Journal of Mathematical Psychology 51 (6), 343–358.

Karatzoglou, A., Smola, A., Hornik, K., Zeileis, A., 2004. kernlab - an s4 package for kernel methods in r. Journal of Statistical Software 11 (9), 1–20.

Kim, Y., Street, W. N., Menczer, F., 2002. Meta-evolutionary ensembles. In: Proceeding of the 2002 International Joint Conference on Neural Networks, Vols 1-3. Ieee, New York, pp. 2791–2796.

Langley, P., 2000. Crafting papers on machine learning. In: Langley, P. (Ed.), Proceedings of 17th International Conference on Machine Learning. Morgan Kaufmann, Stanford CA, pp. 1207–1216.

Larivière, B., Van den Poel, D., 2005. Predicting customer retention and profitability by using random forests and regression forests techniques. Expert Systems with Applications 29 (2), 472–484.

Leisch, F., Dimitriadou, E., 2012. R package mlbench: Machine learning benchmark problems.

Li, Q., Racine, J., 2007. Nonparametric Econometrics: Theory and Practice. Princeton University Press, Princeton.

Liaw, A., Wiener, M., 2002. Classification and regression by randomForest. R News 2 (3), 18–22.

Luo, T., Kramer, K., Goldgof, D., Hall, L., Samson, S., Remsen, A., Hopkins, T., 2004. Recognizing plankton images from the shadow image particle profiling evaluation recorder. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 34 (4), 1753 –1762.

Merkle, E. C., Shaffer, V. A., Feb. 2011. Binary recursive partitioning: Background, methods, and application to psychology. British Journal of Mathematical & Statistical Psychology 64 (1), 161–181.

Nemenyi, P., 1963. Distribution-free multiple comparisons. Ph.D. thesis, Princeton University.

Park, J. I., Liu, L., Ye, X. P., Jeong, M. K., Jeong, Y.-S., Jan. 2012. Improved prediction of biomass composition for switchgrass using reproducing kernel methods with wavelet compressed FT-NIR spectra. Expert Systems with Applications 39 (1), 1555–1564.

Prinzie, A., Van den Poel, D., Apr. 2008. Random forests for multiclass classification: Random MultiNomial logit. Expert Systems with Applications 34 (3), 1721–1732.

Provost, F., Fawcett, T., Kohavi, R., 1998. The case against accuracy estimation for comparing induction algorithms. In: Shavlik, J. (Ed.), Machine Learning. Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98). Morgan Kaufmann Publishers, Madison, WI, USA, pp. 445–453.

R Core Team, R., 2012. R installation and administration. Manual Version 2.15.1.

Schölkopf, B., Smola, A., 2002. Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA.

Shawe-Taylor, J., Cristianini, N., 2004. Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge, UK.

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T., 2009. ROCR: visualizing the performance of scoring classifiers.

Sylvester, J., Chawla, N., 2005. Evolutionary ensembles: Combining learning agents using genetic algorithms. In: AAAI Workshop on Multi-agent Learning. pp. 46–51.

Thorleuchter, D., Van den Poel, D., Dec. 2012. Predicting e-commerce company success by mining the text of its publicly-accessible website. Expert Systems with Applications 39 (17), 13026–13034.

Üstün, B., Melssen, W. J., Buydens, L. M. C., Mar. 2006. Facilitating the application of support vector regression by using a universal pearson VII function based kernel. Chemometrics and Intelligent Laboratory Systems 81 (1), 29–40.

Webb, G. I., Aug. 2000. MultiBoosting: a technique for combining boosting and wagging. Machine Learning 40 (2), 159–196.

Willighagen, E., 2012. R package genalg: R based genetic algorithm.

Yao, X., Liu, Y., Jun. 1998. Making use of population information in evolutionary artificial neural networks. Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics 28 (3), 417–425.

Zhou, Z. H., Wu, J. X., Tang, W., May 2002. Ensembling neural networks: Many could be better than all. Artificial Intelligence 137 (1-2), 239–263.

3 CRM in Social Media: Predicting Increases in Facebook Usage Frequency

Ballings, M, Van den Poel D. CRM in Social Media: Predicting Increases in Facebook Usage Frequency. Submitted for peer review in 2013 to European Journal of Operational Research.

3.1 Abstract

The purpose of this study is to (1) assess the feasibility of predicting increases in Facebook usage frequency, (2) evaluate which algorithms perform best, (3) and determine which predictors are most important and describe their relationship to the response. We benchmark the performance of Logistic Regression, Random Forest, Stochastic Adaptive Boosting, Kernel Factory, Neural Networks and Support Vector Machines using five times twofold cross-validation. The results indicate that it is feasible to create models with high predictive performance. The top performing algorithm was Stochastic Adaptive Boosting with a cross-validated AUC of 0.66 and accuracy of 0.74. The most important predictors include deviation from regular usage patterns, frequencies of likes of specific categories and group memberships, average photo album privacy settings, and recency of comments. Facebook and other social networks alike could use predictions of increases in usage frequency to customize its services such as pacing the rate of advertisements and friend recommendations, or adapting News Feed content altogether. The main contribution of this study is that it is the first to assess the prediction of increases in usage frequency in a social network.

3.2 Introduction

With 1.23 billion active monthly users the social network site Facebook has grown one of the world's largest user bases (Facebook, 2014). Of those monthly users 62% uses the site on a daily basis (757 million) (Facebook, 2014). While Facebook's service is free, its business model is based on advertising. The more time users spend on the platform, the more page impressions Facebook can sell to advertisers. Revenues are directly determined by the number of active users and the time that they spend on the network. Hence, increasing user activity and as such generating ad impressions and clicks is one of Facebook's primary objectives (Claussen et al., 2013).

One way of achieving this objective is predicting increases in usage frequency. More specifically, the binary problem of usage increase[1] prediction consists in predicting whether a user is going to increase usage of the product or service. If a user is predicted not to increase usage, appropriate actions can be taken. One possible action is to propose friendships with unknown users, in addition to the usual practice of proposing friendships with people the user is likely to know in real life (e.g., friends of friends). These new friendships will drive new content to the user's personalized on-line newspaper called News Feed, and could provide a stimulation for increased usage.

Together with acquisition, cross-sell and retention modeling, usage increase modeling (up-sell) shapes the field of predictive modeling in analytical Customer Relationship Management (aCRM) (Ngai et al., 2009). To the best of our knowledge no aCRM study has been attempted in the social media industry. This leaves several questions unanswered such as (1) is predicting usage increases even feasible (i.e., is predictive performance high enough), (2) which algorithms perform best on these data and, (3) which predictors are most important and how are they related to the response variable? This study aims to fill this gap in literature by conducting an empirical study using Facebook data.

The remainder of this article is organized as follows. In the second section of this manuscript, we highlight our contribution by providing a literature review of aCRM studies per industry. Third, the data, time window, variables, algorithms, assessment criteria, partial dependence plots, and cross-validation will be detailed in the methodology section. Fourth, we discuss the results. The fifth section concludes this paper. The penultimate section discusses the managerial implications. In the seventh and final section we discuss the limitations and directions for future research.

[1]Usage frequency increase prediction is equivalent to the more widely known up-sell prediction. Because Facebook is a free service, the term usage increase is more appropriate than the term up-sell.

3.3 Literature Review

There are two strategies to usage increase management: untargeted and targeted management (Burez and Van den Poel, 2007). An untargeted strategy relies on mass advertising to increase usage such as promoting new features and functionalities. In contrast, a targeted strategy comprises (1) identifying which users or customers are (not) going to increase usage and subsequently (2) taking appropriate actions to counter or facilitate this process (Hung et al., 2006). Two targeted strategies exist: reactive and proactive. A reactive approach consists in waiting until a user decreases usage and subsequently trying to reverse the trend. In a proactive approach the company tries to predict whether users are going to decrease usage in advance. If a user is predicted to decrease usage appropriate actions can be taken such as reducing advertisements or recommending new friends. Analogously, if a user is predicted to increase usage, more advertisements could be directed to the user and friend recommendations could be paced and saved for later when needed.

As indicated, targeted proactive approaches can have many advantages but they can be very wasteful if predictions are inaccurate (Burez and Van den Poel, 2007). Hence the goal is to predict usage increases as accurately as possible. This study focuses on illustrating the feasibility of usage increase modeling in a major social network, Facebook. Specifically, this research uses data mining techniques to find the best predictive model and identify top predictors. This will allow the social network to increase usage of its services.

To highlight our contribution Table 3.1 provides an extensive literature review of predictive aCRM studies per industry. Popular industries are financial and insurance services, telecommunications and retail. To the best of our knowledge no other study has investigated the feasibility of usage increase modeling (i.e., up-sell modeling), or any other application for that matter, in the social media industry. It is important to note that Table 3.1 only contains aCRM studies. The search strategy for Table 3.1 is based on the requirement that all studies have to have a prediction focus aimed at one-to-one targeting of users or customers (as is the case with aCRM). While research on social relationships in Facebook has also studied users' connectedness with the Facebook platform it focuses on emotional connectedness to Facebook and the degree to which Facebook is integrated into individuals' daily lives (Clayton et al., 2013) using surveys. For this purpose Ellison et al. (2007) developed a survey scale, called the Facebook Intensity scale. Whereas aCRM focuses on large scale operational deployments of prediction models for one-to-one targeting, the social relationship research stream focuses mainly on description models for theory development and creating strategic insights and is therefore out of the scope of Table 3.1.

Table 3.1: Predictive aCRM studies per industry

Industry	Studies
Fundraising	Verhaert and Van den Poel (2011)
Retail (Supermarket)	Migueis et al. (2012b)
	Migueis et al. (2012a)
	De Bock and Van den Poel (2012)
	Burez and Van den Poel (2009)
	Buckinx and Van den Poel (2005)
Publishing (Newspaper)	Ballings and Van den Poel (2012)
	Coussement and Van den Poel (2009)
	Burez and Van den Poel (2009)
	Coussement and Van den Poel (2008)
Automobiles	Baecke and Van den Poel (2013)
Home Appliances	Prinzie and Van den Poel (2008)
Financial and insurance services	Glady et al. (2009)
	De Bock and Van den Poel (2012)
	De Bock and Van den Poel (2011)
	Burez and Van den Poel (2009)
	Prinzie and Van den Poel (2006)
	Van den Poel and Lariviere (2004)
	Larivière and Van den Poel (2005)
	Benoit and Van den Poel (2012)
	Xie et al. (2009)
	Smith et al. (2000)
	Eiben et al. (1998)
	Kumar and Ravi (2008)
	Prinzie and Van den Poel (2011)
DIY Supplies	De Bock and Van den Poel (2012)
	De Bock and Van den Poel (2011)
	Baesens et al. (2004)
Internet service provider	Madden et al. (1999)
Telecommunications	Lemmens and Croux (2006)
	De Bock and Van den Poel (2011)
	Burez and Van den Poel (2009)
	Mozer et al. (2000)
	Kim (2006)
	Bolton (1998)

	De Bock and Van den Poel (2012)
	Weerahandi and Moitra (1995)
	Datta et al. (2000)
	Hung et al. (2006)
	Wei and Chiu (2002)
	Au et al. (2003)
	Hwang et al. (2004)
	Neslin et al. (2006)
	Lima et al. (2009)
	Verbeke et al. (2011)
Mail order	De Bock and Van den Poel (2012)
	De Bock and Van den Poel (2011)
	Thorleuchter et al. (2012)
Pay TV	Lemon et al. (2002)
	Burez and Van den Poel (2008)
	Burez and Van den Poel (2007)
	Burez and Van den Poel (2009)
E-commerce	Van den Poel and Buckinx (2005)
Social Media	This study

The social media industry is different from the industries listed in Table 3.1 in that much more variables are available. While predictive models are usually based on administrative, operational, or complaints data, profiles on social media contain geographical (e.g., hometown, current town, check-ins), demographical (e.g., birthday, gender), professional (e.g., job function, company), social (e.g., friends, family), and personal (e.g., interests, likes, political views, pictures, videos, albums) information. Hence empirically testing the feasibility of usage increase modeling and assessing which variables are most important in social media is a relevant undertaking.

Besides evaluating the importance of specific variables and describing their relationship to the response, it is key to understand which algorithms perform best. This study benchmarks the predictive performance of six algorithms that have proven to be top performers in literature: Logistic Regression (Berkson, 1944), Random Forest (Breiman, 2001), Stochastic Boosting (Friedman, 2001), Support Vector Machines (Cortes and Vapnik, 1995), Kernel Factory (Ballings and Van den Poel, 2013a), and Artificial Neural Networks (McCulloch and Pitts, 1943).

3.4 Methodology

3.4.1 Data and Time Window

To be able to extract data from Facebook user profiles we developed a Facebook application. In order to stimulate participation we offered a prize. When a user ran the application he or she was presented with an authorization box, which specified the data that were being collected. Immediately after giving permission, the application extracted the data. Next, users were presented with some questions that allowed us to determine the winner of the prize. The data were collected between May 13, 2012 and January 22, 2013.

Selection effects could occur when a user chooses to run the application. We took steps to mitigate these effects by targeting a recruitment campaign (i.e., Facebook advertisements) towards a representative sample of Facebook users. Confirming whether the sample is representative of the general Facebook population is challenging in that Facebook does not publish demographic statistics of its users. However, it is possible to obtain some official gender and age statistics through the use of Facebook's advertisement targeting system. The extraction of these statistics is only possible for 25 countries, hence we chose the top 25 countries in terms of the absolute number of users. A χ^2 test indicates that our sample is not representative of the top 25 countries on both age and gender with respectively $\chi^2(5) = 237.01$, p<0.001 and $\chi^2(1) = 18.27$, p<0.001. To investigate this further Figure 3.1 compares our recruited sample with the population. In terms of the age variable the sample is somewhat more representative for populations with a larger group of 20-29 year olds. The difference between men and woman is also somewhat more pronounced. Despite the significant differences the demographics of our study sample are to a large degree comparable to the demographic characteristics of the top 25 Facebook countries. Other studies using Facebook data obtained comparable samples (e.g., Aral and Walker, 2011). The ranking of the age groups and genders are equivalent for this study and the Facebook population. One has to keep in mind though that the conclusions that we will draw about for example the feasibility of predicting usage frequency increases are somewhat more representative for populations with a larger group of 20-29 year olds and a higher proportion of males than the top 25 countries. It is important to note that "Facebook is a standardized research instrument" (Lewis et al., 2008). Results from subsequent data analysis are "formally replicable in a way most 'case study' data are not" (Lewis et al., 2008) and given that our study is the first to compile and analyze such a rich data set, we believe the results are very valuable.

The main deliverable of this research is an individual-level model to target a user, yes or no. An unambiguous conclusion for each user needs to be made about his or her future behavior (Buckinx and Van den Poel, 2005). A natural approach is to build a binary classification model to classify a user as either a target or not. This methodology is often used in literature on partial churning (e.g., Buckinx and Van den Poel, 2005; Migueis et al., 2012a,b). The threshold of change in usage level, to classify a user or customer, is an input parameter given by the company's management. The threshold should be rather high in order to target only the users or

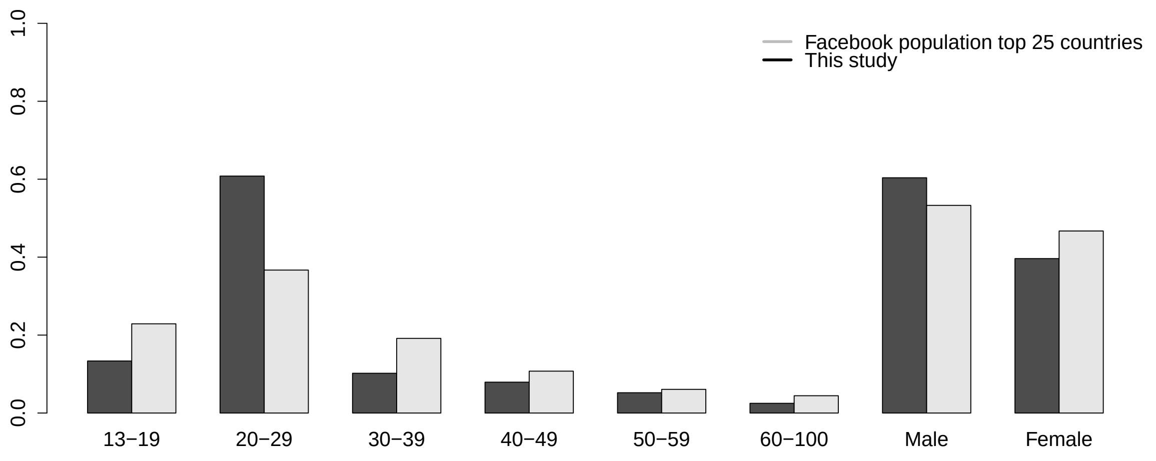


Figure 3.1: Comparison of study sample and demographic characteristics of top 25 Facebook countries

customers most prone to display the target behavior. A common threshold is 40% (see Buckinx and Van den Poel, 2005; Migueis et al., 2012a,b). Because the sample is relatively small and to avoid an imbalanced response variable we decided to select 25% as a threshold to classify the users in our sample. To determine usage increase we needed the users to run the application a second time at a later stage. If from the independent to the dependent period (see Figure 3.2) usage increased by 25% we consider a user to increase usage. Usage is defined as the sum of the following user actions: likes, status updates, videos uploads, photo uploads, albums created, posted links, check-ins, posted notes and comments made. The final sample contains 921 Facebook users of which 24.6% were determined to increase usage in the dependent period[2]. We used SAS 9.3 and R 3.0.1 (R Core Team, 2013) to perform the necessary data preparations and statistical modeling.

3.4.2 Predictors

Table 3.2 provides an overview of the 418 predictors used in this study. We mined as much information as possible from the users' profiles. The predictors can be classified in the following categories: demographic and identification variables, geographical variables, professional and educational variables, social variables, personal variables, general Facebook account variables, likes, statuses, photos, videos, albums, events, links, check-ins, notes, games, tags, comments made and comments received. In Table 3.2 IND is short for INDicator and resolves to 1 if the profile's field is used, otherwise it resolves to 0. REC is short for RECency and is the elapsed time since an event. SDIET is short for Standard Deviation Inter- Event Time. MIET is short for Mean Inter- Event Time. Any reference to 'like' in this study, unless otherwise

[2]We assessed the sensitivity of the performance of the benchmarked algorithms to two more cut-offs: 15% and 35%. These cut-offs result in respectively 25.9% and 22.8% of the sample to be classified as having increased usage.

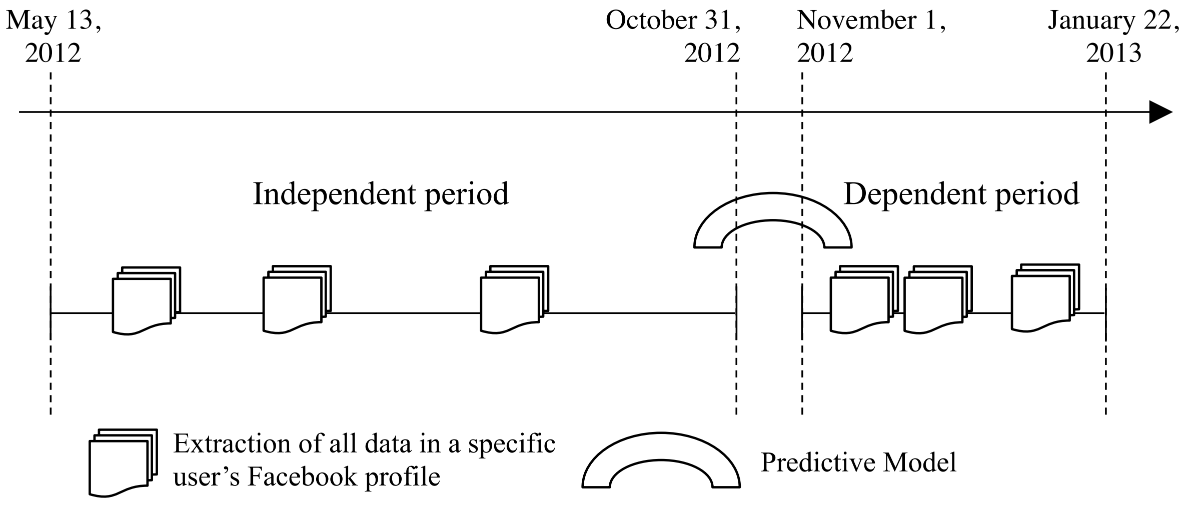


Figure 3.2: Time window

stated, is not a 'like' of content generated by another user (e.g., a status update) but of a page, band, leisure, app, etc. The variable username refers to whether the second part of a user's URL is upgraded to a name (created by the user) (the default is a numeric identifier created by Facebook). The term 'user tags' refers only to tags of the user himself/herself. In the case of photos and videos, the word 'created' refers to uploaded or created, and immediately uploaded, with the Facebook app. Facebook only allows extraction of the 25 most recent status updates, photo uploads, link uploads, album uploads, check-ins, video uploads and notes. To handle this problem we computed frequency by time as to no user in our database reaches these limits. Frequency (COUNT) of status updates, photo uploads and link uploads is for the last 7 days, album uploads and check-ins is for the last 4 months, and video uploads and notes is for the last year.

Table 3.2: Overview of predictors

Variable category	Variable
Demographic and identification variables	Age
	IND(gender)
	IND(gender== female)
	IND(email)
	IND(website)
Geographical variables	IND(hometown)
	IND(location)
Professional/ Educational variables	COUNT(languages)
	COUNT(work)
	COUNT(educations)
	IND(type==High School)

	IND(type==College)
	IND(type==Graduate School)
Social variables	COUNT(family)
	IND(interested in male)
	IND(interested in female)
	IND(sexual preference heterosexual)
	IND(relationship status)
	IND(relationship status==Engaged)
	IND(relationship status==In a relationship)
	IND(relationship status==It's complicated)
	IND(relationship status==Married)
	IND(relationship status==Single)
	COUNT(OF 23 family relationship types) (e.g., aunt)
	COUNT(Friend connections)
	COUNT(Groups)
Personal variables	COUNT(favorite teams)
	COUNT(sports)
	COUNT(television)
	COUNT(music)
	COUNT(movies)
	COUNT(books)
	COUNT(activities)
	COUNT(inspirational people)
	COUNT(interests)
	COUNT(OF 10 television categories) (e.g., Show)
	COUNT(activity category==Interest)
	COUNT(activity category==Sport)
	COUNT(activity category==Athlete)
	COUNT(activity category==Non-profit)
	IND(OF 14 interests) (e.g., Design)
	IND(OF 23 sports) (e.g., Fitness)
	IND(bio)
	IND(quotes)
	IND(political)
	IND(religion)
General Facebook Account variables	Length Facebook membership
	Recency last update=REC(profile update created)
	MEAN(album privacy==custom)
	MEAN(album privacy==everyone)

	MEAN(album privacy==friends)
	MEAN(album privacy==friends-of-friends)
	MEAN(album privacy==networks)
	Profile completeness=SUM(IND(37 profile variables))
	IND(username)
	Time ratio=SDIET(all actions)/MIET(all actions)
Likes	COUNT(OF 188 like categories) (e.g., Musician/band)
	COUNT(likes)
	COUNT(check-in likes)
	REC(like created)
	MIET(like created)
	SDIET(like created)
	COUNT(statuses likes)
	COUNT(photos likes)
	COUNT(albums likes)
	COUNT(check-in likes)
Statuses	COUNT(statuses)
	REC(status updated)
	MIET(status updated)
	SDIET(status updated)
Photos	COUNT(photos)
	REC(photo created)
	MIET(photo created)
	SDIET(photo created)
Videos	COUNT(videos)
	REC(video created)
	MIET(video created)
	SDIET(video created)
Albums	COUNT(albums)
	REC(album created)
	MIET(album created)
	SDIET(album created)
Events	COUNT(events)
	COUNT(event rsvp status==attending)
	COUNT(event rsvp status==unsure)
	MIET(event created)
	SDIET(event created)
Links	COUNT(links)
	REC(link created)

	MIET(link created)
	SDIET(link created)
Check-ins	COUNT(check-ins)
	REC(check-in created)
	MIET(check-in created)
	SDIET(check-in created)
	IND(check in app==Facebook for iPhone)
	IND(check in app==Facebook for Android)
	IND(check in app==BlackBerry App)
	IND(check-in app==Facebook for iPad)
	IND(check-in app==Mobile)
Notes	COUNT(notes)
	REC(note created)
	SDIET(note created)
	MIET(note created)
Games	COUNT(games)
	REC(game created)
	MIET(game created)
	SDIET(game created)
Tags	REC(photo user tags)
	COUNT(video user tags)
	COUNT(photo user tags)
	COUNT(check-in user tags)
	MIET(photo user tags)
	SDIET(photo user tags)
	REC(video user tags)
	MIET(video user tags)
	SDIET(video user tags)
Comments made	REC(videos comments)
	REC(photos comments)
	REC(albums comments)
	REC(statuses comments)
	REC(links comments)
	REC(check-ins comments)
	MIET(videos comments)
	SDIET(videos comments)
	MIET(photos comments)
	SDIET(photos comments)
	MIET(albums comments)

	SDIET(albums comments)
	MIET(statuses comments)
	SDIET(statuses comments)
	MIET(links comments)
	SDIET(links comments)
	MIET(check-ins comments)
	SDIET(check-ins comments)
	COUNT(videos comments)
	COUNT(photos comments)
	COUNT(albums comments)
	COUNT(statuses comments)
	COUNT(links comments)
	COUNT(check-ins comments)
Comments received	REC(videos comments received)
	REC(photos comments received)
	REC(albums comments received)
	REC(statuses comments received)
	REC(links comments received)
	REC(check-ins comments received)
	MIET(videos comments received)
	SDIET(videos comments received)
	MIET(photos comments received)
	SDIET(photos comments received)
	MIET(albums comments received)
	SDIET(albums comments received)
	MIET(statuses comments received)
	SDIET(statuses comments received)
	MIET(links comments received)
	SDIET(links comments received)
	MIET(check-ins comments received)
	SDIET(check-ins comments received)
	COUNT(videos comments received)
	COUNT(photos comments received)
	COUNT(albums comments received)
	COUNT(statuses comments received)
	COUNT(links comments received)
	COUNT(check-ins comments received)

3.4.3 Algorithms

This section will cover our choices for the parameters of the classification algorithms used in this study. In the remainder of this study the acronyms LR, RF, AB, KF, NN and SV respectively denote Logistic Regression, Random Forest, AdaBoost, Kernel Factory, Neural Network, and Support Vector Machines.

3.4.3.1 Logistic Regression

In order to avoid overfitting we use the lasso approach to regularized Logistic Regression. Lasso imposes a bound on the sum of the absolute values of the coefficients. As such coefficients are shrunk towards zero (Guisan et al., 2002). We determine the shrinkage parameter by cross validation. To fit the model the *glmnet* R package is used (Friedman et al., 2010, 2013). The α parameter is set to one to obtain the lasso method and we let the function compute the sequence of λ by setting *nlambda* to 100 (the default).

3.4.3.2 Random Forest

Random Forest requires two parameters: the number of variables to try at each split and the number of trees in the ensemble. We follow Breiman's recommendation (Breiman, 2001) and set the number of variables to the square root of the total number of predictors and use a large number of trees (500). To create the model the *randomForest* R package (Liaw and Wiener, 2012, 2002) is used.

3.4.3.3 Stochastic AdaBoost

Initial implementations of boosting (Freund and Schapire, 1996) are considered an approximation to deterministic weighting (Friedman, 2002). One of the most recent boosting variants is stochastic boosting and improves on the original algorithms by incorporating randomness as an integral part of the procedure (Friedman, 2002). Two important parameters are the number of terminal nodes in the base classifiers and the number of iterations. We set the maximum number of nodes to eight by setting the maximum depth of the trees to three which is in line with the recommendations of Friedman (2002). In addition we set the number of iterations to 500. Stochastic boosting is implemented with the *ada* R package (Culp et al., 2012).

3.4.3.4 Support Vector Machines

An important parameter in Support Vector Machines (SVM) is the kernel function (Martin-Barragan et al., 2014). The most popular choices are the linear, polynomial, and radial basis (RBF) kernel (Ballings and Van den Poel, 2013a). In this study we choose the RBF because of the following reasons: (1) it can model non-linear relationships whereas the linear kernel cannot, (2) it has less hyperparameters than the polynomial function, and (3) it has less computational problems (such as numeric overflow) because the kernel values are bound by zero and one, while the polynomial kernel value may go to infinity (Coussement and Van den Poel, 2008). The RBF kernel requires the choice of only one hyperparameter γ, the width of the Gaussian (Ben-Hur and Weston, 2010). In addition the SVM penalty parameter *C*, also called the cost or soft margin constant, specifies the trade-off between hyperplane violations and the size of the margin. One can not know in advance which *C* and γ are best for a given problem. We follow the recommendation of Hsu et al. (2010) to perform a grid search on *C*= $[2^{-5}, 2^{-4}, ..., 2^{15}]$ and $\gamma = [2^{-15}, 2^{-13}, ..., 2^{3}]$ to identify the best combination. To map the decision values to probabilities we used Platt's method (Platt, 2000). Support Vector Machines are implemented through the *svm* function of the *e1071* R package (Meyer et al., 2012).

3.4.3.5 Kernel Factory

Ballings and Van den Poel (2013a) recommend the *burn* method for Kernel Factory. This method automatically selects the best kernel function. Furthermore, we use the recommended values for the number of column partition and row partitions (Ballings and Van den Poel, 2013b). Kernel Factory is implemented through the *kernelFactory* R package (Ballings and Van den Poel, 2013b).

3.4.3.6 Neural Network

The final base classifier is a feed-forward artificial neural network optimized by BFGS which is vastly more efficient, reliable and convenient than backpropagation. We use one layer of hidden neurons which is generally sufficient for classifying most data sets (Dreiseitl and Ohno-Machado, 2002). Before applying the neural network we rescale the numerical predictors to [0,1]. The binary variables are left untouched {0,1}. Scaling of the data is necessary to overcome numerical problems and to obtain training efficiency. The algorithm is implemented using the *nnet* R package (Ripley, 2013; Venables and Ripley, 2002). The network weights at the start of the iterative procedure are chosen at random (Ripley, 1996, pg. 154), hence results of subsequent runs of *nnet* can differ. This mimics a newborn's brain: developed but without any real knowledge (Venkatesh et al., 2014). The *entropy* parameter is set to use the maximum conditional likelihood as recommended by Spackman (1991) and Ripley (1996, pg. 149). The *rang* parameter, controlling the range of the initial random weights parameter was left at the default of 0.5. We left the parameters *abstol* and *reltol* to respectively 1.0e-4 and 1.0e-8. We used weight decay to avoid overfitting (Dreiseitl and Ohno-Machado, 2002) and hence the maximum number of weights (*MaxNWts*) and the number of iterations (*maxit*) were set to very large values (5000) in order not to run into a situation of early stopping. Finally, the weight decay factor and the number of nodes in the hidden layer were determined by performing a grid search (Dreiseitl and Ohno-Machado, 2002). We tried all combinations of *decay*={0.001, 0.01, 0.1} (Ripley, 1996, pg. 163), and *size*=[1, 2, ..., 20] (Ripley, 1996, pg. 170) and selected the optimal combination.

3.4.4 Assessment Criteria

To assess the performance of the models we use accuracy and area under the receiver operator characteristic curve (AUC or AUROC). Accuracy is defined as follows:

$$Accuracy = \frac{TP + TN}{P + N} \tag{3.1}$$

with TP: True Positives, TN: True Negatives, P: Positives, N: Negatives

All classifiers were set to probabilistic output because a ranking is needed of which users are most likely to increase usage. To calculate accuracy a threshold then needs to be chosen to determine whether a user is classified as having increased usage or not. This cutoff should be chosen to correspond to the proportion of users that will be targeted (Hand, 2005) and is dictated by the marketing budget. Marketing analysts are often interested in the top 10% of users most likely to display the target behavior (Coussement et al., 2010). Hence, in this study we will use the threshold value that corresponds to a proportion of 10%.

In contrast to accuracy, AUC is insensitive to the cutoff value of the posterior probabilities (Hanley and Mcneil, 1982; Thorleuchter and Van den Poel, 2012) that classifies a Facebook user as either increasing usage or not. More specifically, while accuracy measures the performance of the model at one specific cutoff value, AUC assesses performance across all possible cutoff values. AUC is considered an objective performance metric for classification models (Provost et al., 1998) and is defined as follows (Ballings and Van den Poel, 2013a):

$$AUC = \int_0^1 \frac{TP}{TP + FN} d\frac{FP}{FP + TN} = \int_0^1 \frac{TP}{P} d\frac{FP}{N} \tag{3.2}$$

with TP: True Positives, FN: False Negatives, FP: False Positives, TN: True Negatives, P: Positives, N: Negatives

AUC is bound by 0.5 and 1, where the former value denotes that predictions are not better than random, and the latter indicates perfect prediction. In some cases AUC can be lower than 0.5 on the test sample. This is almost always due to overfitting leading the model to generalize poorly.

3.4.5 Variable importance evaluation

To assess the importance of the predictors in usage increase prediction we use a permutation-based measure. This type of measure is often used in Random Forest and other ensembles but its usefulness is not limited to these algorithms. The decrease in accuracy if variables are permuted is a popular implementation but has the limitation that accuracy is not a good performance measure in imbalanced settings. Janitza et al. (2013) have compared permutation-based decrease in accuracy and AUC and conclude that the AUC- based importance measure is preferred to the standard accuracy- based measure whenever the two response classes have different class sizes. As this is the case in our study (see subsection 3.4.1), we will follow their recommendation and use the decrease in AUC as the variable importance measure in this study.

Janitza et al. (2013) work with out-of-bag data, which is different for all trees. Hence they average the AUC over all trees per ensemble. Since we work with test data we compute the measure for the ensemble. This is essentially the same approach. The importance measures are averaged across the ten folds obtained through five times twofold cross-validation by taking the median.

3.4.6 Partial Dependence Plots

Partial Dependence Plots (Friedman, 2001; Hastie et al., 2009, Section 10.13.2) offer a way to describe the relationship between a predictor and the response variable. The method can be used with any algorithm, including ensemble algorithms which are often discredited for having low interpretability (Friedman and Meulman, 2003, p24). Partial Dependence Plots can be created as follows. For each value in the range of a given variable iteratively impute that given variable for all the users and compute the mean value of the predicted probability. A plot of each of these values and the mean of half the logit of the predicted probabilities then reveals the relationship between the predictor variable and the response variable (Berk, 2008, p222).

3.4.7 Cross Validation

All reported AUCs and accuracies are medians over five replications of twofold cross-validation (5x2f cv) (Dietterich, 1998; Alpaydin, 1999). In each replication all data instances are randomly assigned to one of two parts that are equal in size. Each part is employed as both a training and test set. The entire process results in ten AUCs and accuracies. The same splits are used for all models. As a measure of dispersion we use the inter quartile range.

In order to test for significant differences we follow the suggestions of Demsar (2006) to use Friedman's test with Nemenyi's post hoc test (Nemenyi, 1963) for comparing the classifiers. Classifiers are ranked, within folds, with the top performing classifier receiving rank 1 and the worst performing classifier receiving a rank equal to the number of classifiers (if no ties are observed). Ties are handled by taking the average ranks. By using this approach the relatedness of the folds is incorporated (classifier ranks are calculated per fold and then the average rank is calculated per classifier). In contrast to when the median is computed, they are not treated as independent (Demsar, 2006). In summary, the average ranks of the AUCs preserve the order of the folds while the median of the AUCs does not. This is also the reason why results can sometimes differ to a limited extent (Ballings and Van den Poel, 2013a). Hence average ranks allow a stricter comparison than the median. We report both the average ranks per classifier and the median.

Two classifiers perform significantly different if their average ranks differ by at least the critical difference. The critical difference (CD) is defined as follows (Demsar, 2006):

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{3.3}$$

with q_α the critical value (which can be found in any statistical statistical book) for a given p-value and number of classifiers, *k* the number of classifiers, and *N* the number of folds. In this study the critical difference for a p-value of 0.05, 6 classifiers, 10 folds and critical value of 2.850 equals 2.384.

3.5 Results

3.5.1 Models

The results clearly indicate that predicting increases in usage frequency in the social media industry is a viable strategy with a best median AUC of 0.66 and best median accuracy of 0.74. Figure 3.3 and 3.4 respectively display the median AUC and median accuracy of the cross validation folds. Based on AUC AdaBoost comes out on top followed by Random Forest, Logistic Regression, Kernel Factory, Support Vector Machines and Neural Networks. The same pattern can be observed for accuracy except that Random Forest shares its second place with Support Vector Machines and that Logistic Regression shares its third place with Kernel Factory. In sum, AdaBoost is the optimal choice for our data set[3].

These results are substantiated by the median ROC curves (Figure 3.5) and the median accuracy curves (Figure 3.6). There are no important ROC curve crossings except for Kernel Factory performing somewhat worse for lower false positive rates and Random Forest performing somewhat better on high false positive rates. In a similar fashion the accuracy curves show no important crossings. However, the performance gap between the leading classifier AdaBoost and the rest of the classifiers is more pronounced at higher proportions.

Table 3.3 displays the average ranks and the results of the Friedman significance test. The classifiers are divided into two groups: (1) classifiers that perform statistically worse to the top

[3]The results for the two alternative thresholds largely coincide. For a threshold of 15% and performance measure AUC (5x2f CV) the results are as follows (in decreasing order): AB=0.6452, RF=0.6091, LR=0.5985, SV=0.5748, KF=0.5738, and NN=0.5379. For the performance measure accuracy (5x2f CV) the results are as follows (in decreasing order): AB=0.7348, RF=0.722, LR & KF =0.7137, NN=0.7109, and SV=0.7065. Both AdaBoost and Random Forest are again the top performers.

For a threshold of 35% and performance measure AUC (5x2f CV) the results are as follows (in decreasing order): AB=0.6408, RF=0.6040, LR=0.6009, KF=0.5797, SV=0.5711, and NN=0.5147. For the performance measure accuracy (5x2f CV) the results are as follows (in decreasing order): AB=0.7570, LR=0.7375, SV=0.7354, RF=0.7307, KF=0.7242, and NN=0.7166. The results based on AUC are identical to when a cut-off of 25% is used. When accuracy is used as a performance measure, Random Forest has to give way to Logistic Regression and Support Vector Machines.

In sum, the sensitivity analysis clearly indicates that the results are not very sensitive to the chosen cut-off. The results and conclusions are very similar when other thresholds are used. For all tested cut-offs the top performer is AdaBoost.

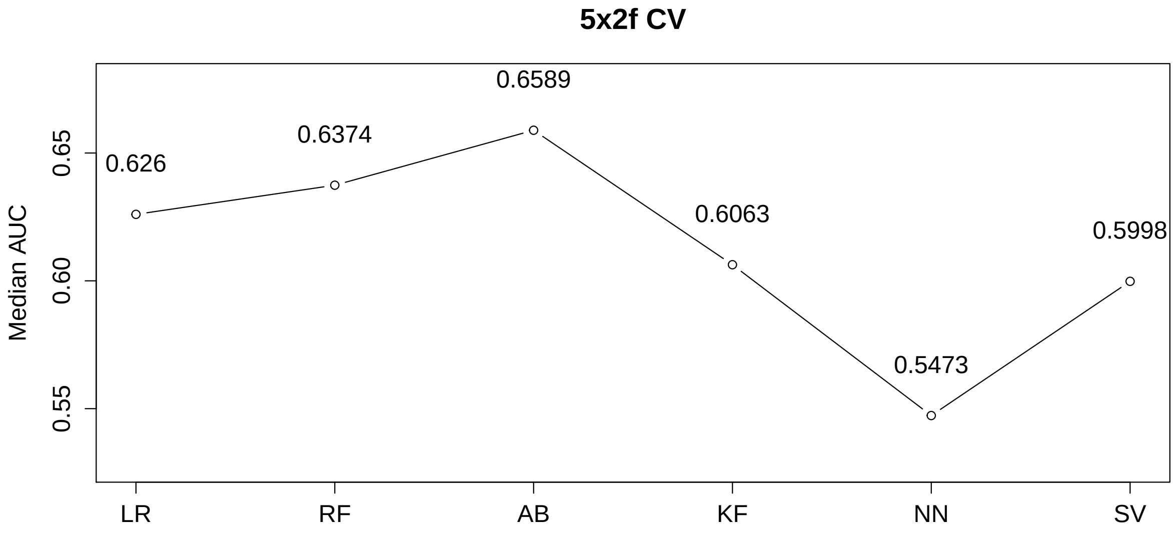


Figure 3.3: Cross validated AUC. LR=Logistic Regression, RF=Random Forest, AB=AdaBoost, KF=Kernel Factory, NN=Neural Network, SV=Support Vector Machines

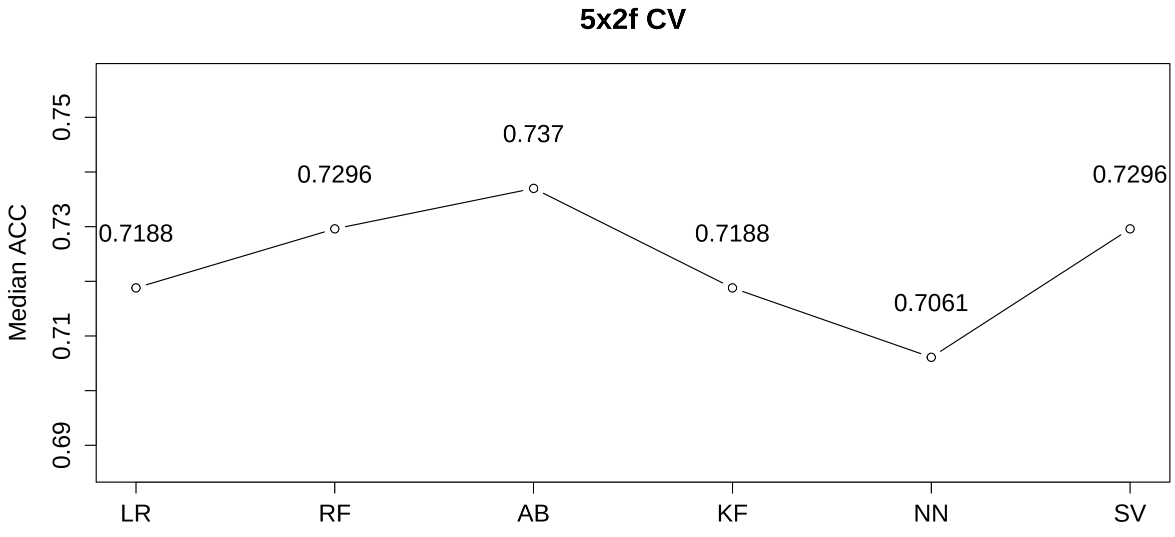


Figure 3.4: Cross validated Accuracy for a cutoff corresponding to the top 10%. LR=Logistic Regression, RF=Random Forest, AB=AdaBoost, KF=Kernel Factory, NN=Neural Network, SV=Support Vector Machines

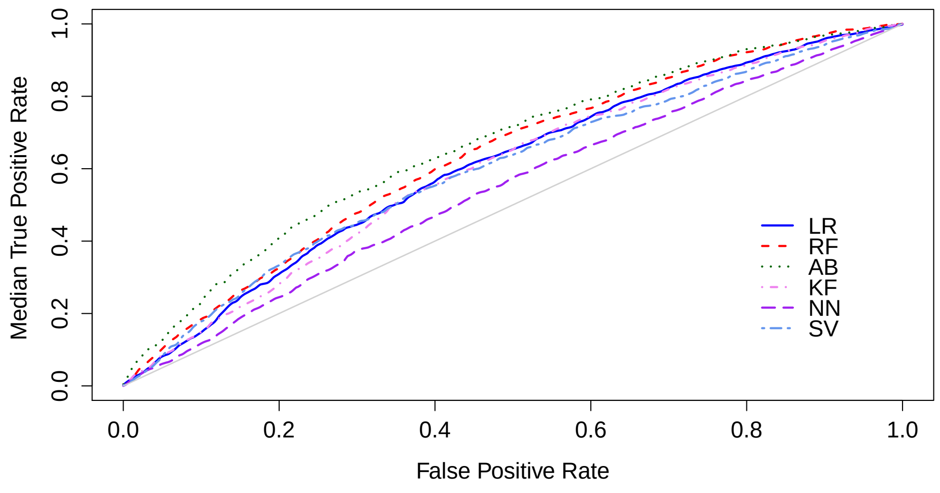


Figure 3.5: Cross validated ROC. LR=Logistic Regression, RF=Random Forest, AB=AdaBoost, KF=Kernel Factory, NN=Neural Network, SV=Support Vector Machines

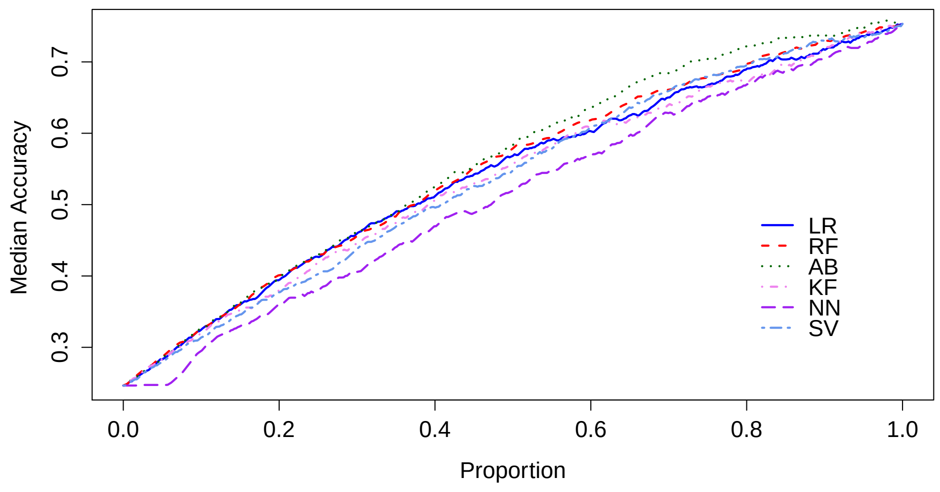


Figure 3.6: Cross validated Accuracy Curve. LR=Logistic Regression, RF=Random Forest, AB=AdaBoost, KF=Kernel Factory, NN=Neural Network, SV=Support Vector Machines

performer (AdaBoost) and (2) classifiers that perform statistically equal to AdaBoost (displayed in boldface in Table 3.3).

Table 3.3: Average ranks

	LR	RF	AB	KF	NN	SV	Friedman $chi^2(5)$
AUC	**3.20**	**2.50**	**1.50**	4.30	5.50	4.00	28.51, $p < 0.001$
Accuracy	**3.75**	**2.75**	**1.70**	4.25	5.65	**2.90**	27.51, $p < 0.001$

Based on AUC, Logistic Regression and Random Forest perform statistically equally well as AdaBoost. Based on accuracy Support Vector Machines does also perform equally well as AdaBoost. Kernel Factory and Neural Networks perform significantly worse than AdaBoost. These results indicate that Random Forest may be a good option whenever speed is an issue. The underlying reason is that Random Forest can be executed in parallel and AdaBoost is inherently sequential.

The feasibility of usage increase prediction is also confirmed when we look at the stability of the results. The inter quartile ranges (IQRs) are displayed in Table 3.4. The IQRs are low for all classifiers, meaning that all classifiers produce stable results.

Table 3.4: Inter Quartile Ranges of the AUCs and accuracies obtained through 5x2f cv

	LR	RF	AB	KF	NN	SV
AUC	0.085	0.016	0.030	0.040	0.026	0.006
Accuracy	0.013	0.012	0.022	0.017	0.011	0.015

There are several possible reasons as to why AdaBoost and Random Forest are so strong in this case. First of all, both methods use decision trees. Trees outperform Logistic Regression if data are not normally distributed (which is often the case in real life data sets) (King et al., 1995) and if data are non-linear. Trees even perform very well on data with extreme distributions (King et al., 1995). Similarly, Neural Networks will only result in the best possible linear combination of weight estimates if at least normality assumptions are met (Bishop, 2002; Matignon, 2005, p.8). Our analyses suggest that indeed normality assumptions are not met (as is the case in many real life data sets). Kernel- based methods then, such as Support Vector Machines and Kernel Factory, are very sensitive to the choice of the kernel function and will either work very well, or very poorly depending on the data (and hence the right kernel) (Ballings and Van den Poel, 2013b). Given the large variety of variables (counts, standard deviations, means, indicators; see Table 3.2) it is highly likely that not all variables conform to one single kernel function. Kernels force the data to a novel data representation in that data are not represented individually anymore (per variable), but through a set of pairwise comparisons or similarities (Vert et al., 2004). This could be a possible reason why trees outperform kernel-based methods in this case. Moreover, the high performance of AdaBoost and Random Forest is not only due to their tree base- classifiers. Random Forest further improves upon the performance of single trees by reducing the variance of the predictions (Bauer and Kohavi, 1999). Boosting improves trees by reducing both bias and variance (Bauer and Kohavi, 1999). This makes AdaBoost and Random Forest all-round high performers. An extensive analysis of why AdaBoost and Random Forest outperform the other methods is out of the scope of this study. However, this paragraph offers plausible possible explanations of why AdaBoost and Random Forest outperform the other methods.

3.5.2 Predictors

To investigate which predictors are driving predictive performance we made a scree plot (Figure 3.7). In the scree plot the variables are sorted in descending order by the five times twofold cross-validated median decrease in AUC of the AdaBoost model when predictors are permuted. Predictors ranked lower than 50 add little to the predictive performance. To demonstrate this we included the top 200 variables in the plot (Figure 3.7). The top 50 predictors are listed in Table 3.5.

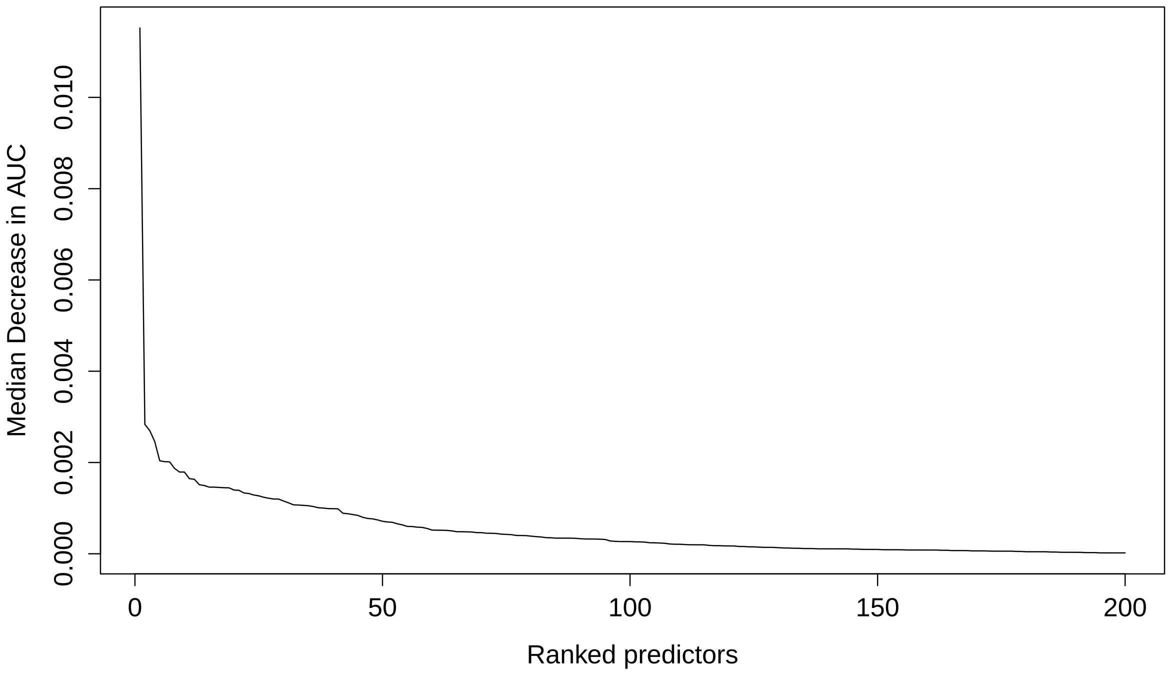


Figure 3.7: Scree plot of importance of 200 top predictors

Table 3.5: Variable importance of top 50 predictors

No.	Variable name	Median Decrease AUC
1	Time ratio	0.01152
2	COUNT(likes from category==Retail and consumer merchandise)	0.00284
3	MEAN(album privacy==custom)	0.00270
4	COUNT(photos)	0.00246
5	SDIET(album created)	0.00204
6	COUNT(Groups)	0.00202
7	COUNT(likes from category==Food/beverages)	0.00201
8	COUNT(likes from category==Public figure)	0.00187
9	REC(links comments received)	0.00179
10	Age	0.00179
11	MIET(video user tags)	0.00165
12	MIET(link created)	0.00163
13	REC(like created)	0.00151
14	SDIET(videos comments)	0.00150
15	COUNT(likes from category==Computers/internet)	0.00146
16	COUNT(likes from category==Shopping/retail)	0.00146
17	COUNT(television category==Tv show)	0.00145
18	COUNT(likes from category==Actor/director)	0.00145
19	COUNT(likes from category==Entertainment)	0.00144
20	COUNT(likes)	0.00140
21	COUNT(music)	0.00139
22	SDIET(event created)	0.00133
23	REC(album created)	0.00132
24	SDIET(status updated)	0.00129
25	COUNT(family relationship==son)	0.00127
26	COUNT(likes from category==Sport)	0.00124
27	Recency last update	0.00122
28	COUNT(likes from category==Cars)	0.00120
29	REC(albums comments received)	0.00120
30	REC(link created)	0.00116
31	COUNT(family relationship==sister)	0.00112
32	SDIET(videos comments received)	0.00107
33	COUNT(likes from category==Movie)	0.00107
34	MIET(links comments received)	0.00106
35	SDIET(links comments received)	0.00105

36	Profile completeness	0.00104
37	SDIET(photos comments)	0.00101
38	SDIET(photo created)	0.00100
39	REC(photo created)	0.00099
40	COUNT(family)	0.00099
41	COUNT(family relationship==nephew)	0.00099
42	REC(note created)	0.00089
43	COUNT(likes from category==Magazine)	0.00088
44	MIET(album created)	0.00086
45	COUNT(activities)	0.00084
46	REC(videos comments)	0.00080
47	COUNT(likes from category==Clothing)	0.00077
48	MIET(photo created)	0.00076
49	COUNT(check-in user tags)	0.00074
50	COUNT(likes from category==Media/news/publishing)	0.00071

The top predictor is the time ratio. Time ratio is the standard deviation of the inter-event time (all events) relative to the mean inter-event time. Hence whether the user's behavior deviates is a strong predictor of usage increase. The frequencies of liking behavior in the categories retail and consumer merchandise, food and beverage and public figures are also in the top 10 predictors. Next to that, photo album privacy settings play an important role. If users invest the time to create custom privacy settings this is predictive of usage increase. In addition, the frequency of photo uploads and the deviation in the creation of albums are important. Other important variables are the number of group memberships, the recency since the user has received comments on his or her content and the user's age. Despite the importance of age, socio-demographics clearly do not play a significant role in the prediction of usage increase. This is in line with other studies in the field of aCRM (Rossi et al., 1996).

Figure 3.8 contains Partial Dependence Plots for a selection of predictors[4] based on the best performing model in our benchmark (AdaBoost). Consider the trend of age. The older a user is, the higher the probability of usage increase. This is not true for users younger than 20: older means lower probability of usage increase. The other relationships are less complex. The number of group memberships has an overall positive relationship with the probability to increase usage. The time ratio (deviation of regular usage in terms of inter event time) has an overall negative relationship with the probability to increase usage. This means that users that have more stable behavioral patterns have a higher probability to increase usage.

[4]The count and time ratio variables do have higher maximum values but these are equal to the value of the maximum predictor value displayed in the plots.

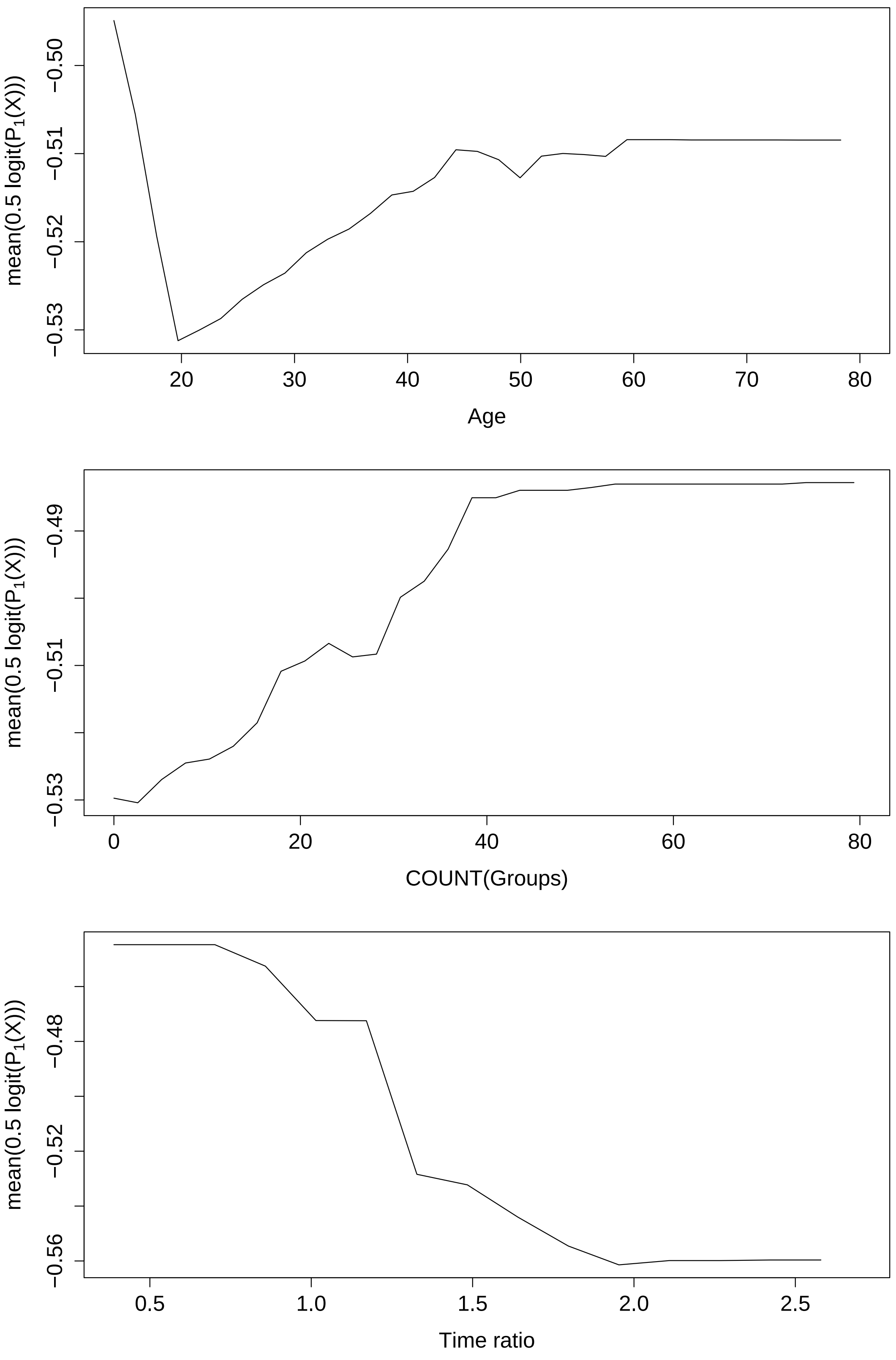


Figure 3.8: Partial Dependence Plots for a selection of predictor variables

3.6 Conclusions

In this study we set out to (1) study the feasibility of usage increase prediction in social media, (2) evaluate which algorithms perform best, (3) and determine the top predictors and describe their relationship with the response.

The results clearly indicate that usage increase prediction is a viable strategy with AUCs up to 0.66 and accuracies up to 0.74. Based on AUC, AdaBoost is the best choice to model usage increase, followed by (in this order) Random Forest, Logistic Regression, Kernel Factory, Support Vector Machines and Neural Networks. A similar pattern can be observed when accuracy is used as a performance measure. AdaBoost is the top performer, with Random Forest and Support Vector Machines sharing a second place, Logistic Regression and Kernel Factory sharing the third place, and Neural Networks on the fourth place. Overall it is clear that AdaBoost is the best choice, closely followed by Random Forest.

The top predictor is the time ratio. This variable is operationalized to capture the user's deviation from his or her regular usage patterns and has been shown to be negatively related to the probability to increase usage. This seems a plausible finding and is consistent with Buckinx and Van den Poel (2005) in that customers (or users) with higher time ratios (i.e., deviations from established behavioral patterns) are less loyal. This can in turn be translated to a lower probability to increase usage. Other important predictors are frequencies of specific liking behaviors, privacy settings of albums, frequencies of photo uploads and deviations in album creations. Groups memberships, recency since receiving comments, and a users' age are also very important. A list of the top fifty predictors is provided in Table 3.5 and serves as a first direction for Facebook Inc. as to which predictors to include in their models. These findings are important for Facebook Inc. and other social media companies for that matter. The following section will discuss the managerial implications.

3.7 Managerial Implications

Facebook's business plan is based on advertising. Therefore, one of Facebook's primary goals is to increase the time users spend on the platform, increase user activity, and generate ad impressions (Claussen et al., 2013). The social network draws users to its platform by delivering social content to users' customized on-line newspapers, called News Feeds. This process is governed by an algorithm with the absolute goal to deliver the right content to the right user at the right time (Facebook, 2013a). Every time someone visits Facebook the News Feed algorithm tries to filter out the posts that are important out of an average of 1500 potential stories from connections and pages (Facebook, 2013a).

The goal with the advertisements that are shown is identical: the right advertisement has to be delivered to the right user at the right time (Facebook, 2013b). Every time a user visits Facebook, the same algorithm chooses between thousands of advertisements to show the most relevant ones (Facebook, 2013b). However, sometimes users might not want to see any advertisements at all. Showing advertisements at that point might alienate users, decrease usage frequency, and decrease time spent on the platform for long periods of time. Once defective behavior has been observed, Facebook could react and turn off advertisements for that particular user. However it may take a while before the user returns to the platform and it even might be too late to recapture the user. Instead of a reactive strategy, we propose a proactive strategy.

More concretely we propose to improve the News Feed algorithm by introducing a predictive component. This will allow Facebook to proactively change tactics and increase revenues. An illustration may clarify our point. Users are attracted if more, new or better social content is delivered to their News Feeds and in contrast they are alienated if more advertisements are delivered. Facebook could employ usage increase prediction as part of the decision process regarding the balance of social content and advertisements. If a user is predicted to increase usage he or she is satisfied with the service (or at least uses the service more than other users) and hence more advertisements can be shown. Friend and page recommendations could be saved for later. In contrast, if a user is predicted not to increase usage, advertisements can be turned off, and new friends and content can be recommended in an attempt to attract and engage the user. Once engaged, advertisements can be turned on again. By proactively adapting its tactics, Facebook can avoid alienating potential unhappy users even further and prevent future defection. At the same time it can count on its satisfied users to create ad impressions and fuel revenue.

The predictive models that we have developed in this paper are viable and could be deployed on a large scale for such a proactive strategy. The News Feed algorithm could be adapted by introducing a switch that (1) turns on advertisements and saves friend and page recommendations for later if the user is predicted to increase usage and (2) turns off advertisements and activates friend and page recommendations if the user is predicted not to increase usage.

3.8 Limitations and directions for Future Research

The first limitation is that we do not have access to full network data (i.e., data about all the connections of the recruited sample). Therefore network effects cannot be included in the analysis. There is a large body of research on social networks (Hellmann and Staudigl, 2014) reporting the influence of network effects on a wide range of behaviors (e.g., Bakshy et al., 2012). These effects are in part driven by homophily, also called endogenous group formation (Hartmann et al., 2008), and (social) influence (Aral et al., 2009). Failure to include network effects in the analysis could bias variable importances. Unfortunately, data of a user's friends are very hard to obtain and we currently do not have these data. However, as Benoit and Van den Poel (2012) note, network- based predictors (e.g., Gómez et al., 2013) may only improve predictions and hence our conclusions about the feasibility of usage increase prediction in social media are substantiated. Future research could try to obtain such data and improve on this study.

The second limitation is selection effects. It might well be possible that the users that are unwilling to share their data (i.e., use our application) may be different from the users in our recruited sample. Web crawling, another method of obtaining Facebook data, also suffers from the limitation that private profiles cannot be crawled (Lewis et al., 2008). Although our approach (using an application with an authorization box) does extract data from users with a private profile, it suffers from the limitation that some of the privacy sensitive users will not be willing to share their data with an application. It can be argued that both groups, users with private profiles and users that don't want to give their data to applications, largely coincide. Although we recognize that this may impact the generalizability of our results we firmly believe that our approach is able to extract data from profiles that cannot be extracted using web crawling and consequently suffers less from generalizability issues than web crawling because of the following reasons. First, setting a profile to private is in part a mechanism for forcing candidate viewers of a user's profile to identify themselves and state their purpose. Candidate viewers that are not deemed trustworthy are not allowed to see a user's profile and become a connection. Hence, we identified ourselves by providing our contact information, and the university's, for questions regarding privacy and stated the purpose of our research. We also informed the user that we encode usernames and that we do not extract personal messages. It can be argued that users may be more inclined to share their data to one application for academic research purposes than to the general, unidentified public. Second, given the massive adoption of Facebook applications that all ask the permission to access a user's profile data, we believe that privacy is of lesser concern in sharing data to applications than it is in sharing data with the general public. Third, we offered an incentive in the form of a prize to share their data. In addition to these arguments, it is important to note that while previous research used a web-crawling approach to extract data from university students (e.g. Lampe et al., 2007; Lewis et al., 2008), with permission of the university in question and Facebook, it is important to note that our approach is more privacy aware, because we ask permission directly from the user. Moreover, Facebook is a standardized research instrument (Lewis et al., 2008) and results from subsequent data analysis are formally replicable in a way most case study data are not (Lewis et al., 2008).

The third limitation of this study is that some of the variables are limited in the number of values. Facebook limits the number of entries per variable that an application can extract to the 25 most recent entries. This limitation has mostly an impact on the frequency variables. To cope with this, we computed the frequency within a specific period of time. We determined the length of this time window per variable as no user in our database reaches the maximum number of 25 entries. The frequency of status updates, photo uploads and link uploads was computed for the last 7 days, album uploads and check- ins for the last 4 months, and video uploads and notes for the last year. An interesting avenue for future work would be to find ways to alleviate this problem. A possible strategy might be to motivate people to revisit the application at different times and to stack the 25 last entries.

The fourth limitation is that private communication data are not included in the analysis. Some work has been done to analyze social relationships in Facebook. Arnaboldi et al. (2013)

study 28 Facebook users and all their relationships and determine the drivers of tie strength. Future research could integrate these insights. Fluctuations in tie strength with specific users may be a valuable predictor in usage increase models. Unfortunately our data do not allow us to incorporate individual communication- based variables. We also expect that these data are very difficult to obtain for a large sample in that users have to authorize data collection (the 28 users in Arnaboldi et al. (2013) are members of the authors' research department).

The fifth limitation is that there might be some bias by not taking seasonality into account (e.g., if part of the users are celebrating Christmas and New year and part are not). It is however, very difficult to automatically detect which user is celebrating New Year and which user is not. This bias does not inflate our results: taking seasonality into account would only increase the predictive performance of our models meaning that we have a conservative result. Our finding that it is feasible to predict usage increase is hence valid but including seasonality would be an interesting avenue for future research.

The sixth limitation is that we do not model website visits or login events. As such we disregard passive users who visit the social network leaving valuable opportunities for advertising untapped. Unfortunately Facebook does not allow to extract visits or login events. If Facebook decides to share these data, an interesting avenue for future research would be to model passive usage and compare it with active usage. We do want to note that the advantage of active usage is that each activity on Facebook instills a user's friends to also visit the website. Therefore it might have a greater impact to target active usage increase.

As a final remark we want to say that although this study has these data-related shortcomings, it is the first aCRM study using such a variety of data. Other published studies in top journals using Facebook data (for other purposes than this study) have similar limitations (e.g., Aral and Walker, 2011). Large investments have been made to create the extractor app and to the best of our knowledge this study is the first to provide insight into the feasibility of predicting usage increases. In sum, we feel that this is a valuable contribution to literature.

3.9 Acknowledgements

We would like to thank Matthias Lelie for his help in this study. Funding for this research was provided by the Special Research Fund (BOF, Bijzonder Onderzoeksfonds), Ghent University, Belgium.

3.10 References

Alpaydin, E., 1999. Combined 5 x 2 cv f test for comparing supervised classification learning algorithms. Neural Computation 11 (8), 1885–1892.

Aral, S., Muchnik, L., Sundararajan, A., Dec. 2009. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. Proceedings of the National Academy of Sciences, 21544 –21549.

Aral, S., Walker, D., Sep. 2011. Creating social contagion through viral product design: A randomized trial of peer influence in networks. Management Science 57 (9), 1623–1639.

Arnaboldi, V., Guazzini, A., Passarella, A., Jun. 2013. Egocentric online social networks: Analysis of key features and prediction of tie strength in facebook. Computer Communications 36 (10-11), 1130–1144.

Au, W. H., Chan, K. C. C., Yao, X., Dec. 2003. A novel evolutionary data mining algorithm with applications to churn prediction. Ieee Transactions on Evolutionary Computation 7 (6), 532–545.

Baecke, P., Van den Poel, D., Aug. 2013. Improving customer acquisition models by incorporating spatial autocorrelation at different levels of granularity. Journal of Intelligent Information Systems 41 (1), 73–90.

Baesens, B., Verstraeten, G., Van den Poel, D., Egmont-Petersen, M., Van Kenhove, P., Vanthienen, J., Jul. 2004. Bayesian network classifiers for identifying the slope of the customer lifecycle of long-life customers. European Journal of Operational Research 156 (2), 508–523.

Bakshy, E., Eckles, D., Yan, R., Rosenn, I., 2012. Social influence in social advertising: Evidence from field experiments. In: Proceedings of the 13th ACM Conference on Electronic Commerce. New York, pp. 146–161.

Ballings, M., Van den Poel, D., 2012. Customer event history for churn prediction: How long is long enough? Expert Systems with Applications 39 (18), 13517–13522.

Ballings, M., Van den Poel, D., 2013a. Kernel factory: An ensemble of kernel machines. Expert Systems with Applications 40 (8), 2904–2913.

Ballings, M., Van den Poel, D., 2013b. R package kernelFactory: an ensemble of kernel machines.

Bauer, E., Kohavi, R., 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36 (1-2), 105–139.

Ben-Hur, A., Weston, J., 2010. A user's guide to support vector machines. Methods in Molecular Biology. Department of Computer Science Colorado State University, pp. 223–239.

Benoit, D. F., Van den Poel, D., Oct. 2012. Improving customer retention in financial services using kinship network information. Expert Systems with Applications 39 (13), 11435–11442.

Berk, R. A., 2008. Statistical Learning from a Regression Perspective. Springer Series in Statistics. Springer.

Berkson, J., Sep. 1944. Application of the logistic function to bio-assay. Journal of the American Statistical Association 39 (227), 357–365.

Bishop, C., 2002. Neural Networks for Pattern Recognition. Oxford University Press.

Bolton, R. N., 1998. A dynamic model of the duration of the customer's relationship with a continuous service provider: The role of satisfaction. Marketing Science 17 (1), 45.

Breiman, L., Oct. 2001. Random forests. Machine Learning 45 (1), 5–32.

Buckinx, W., Van den Poel, D., 2005. Customer base analysis: partial defection of behaviourally loyal clients in a non-contractual FMCG retail setting. European Journal of Operational Research 164 (1), 252–268.

Burez, J., Van den Poel, D., 2007. CRM at a pay-TV company: Using analytical models to reduce customer attrition by targeted marketing for subscription services. Expert Systems with Applications 32 (2), 277–288.

Burez, J., Van den Poel, D., 2008. Separating financial from commercial customer churn: A modeling step towards resolving the conflict between the sales and credit department. Expert Systems with Applications 35 (1–2), 497–514.

Burez, J., Van den Poel, D., 2009. Handling class imbalance in customer churn prediction. Expert Systems with Applications 36 (3), 4626–4636.

Claussen, J., Kretschmer, T., Mayrhofer, P., Mar. 2013. The effects of rewarding user engagement: The case of facebook apps. Information Systems Research 24 (1), 186–200.

Clayton, R. B., Osborne, R. E., Miller, B. K., Oberle, C. D., May 2013. Loneliness, anxiousness, and substance use as predictors of facebook use. Computers in Human Behavior 29 (3), 687–693.

Cortes, C., Vapnik, V., Sep. 1995. Support-vector networks. Machine Learning 20 (3), 273–297.

Coussement, K., Benoit, D. F., Van den Poel, D., Mar. 2010. Improved marketing decision making in a customer churn prediction context using generalized additive models. Expert Systems with Applications 37 (3), 2132–2143.

Coussement, K., Van den Poel, D., 2008. Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. Expert Systems with Applications 34 (1), 313–327.

Coussement, K., Van den Poel, D., 2009. Improving customer attrition prediction by integrating emotions from client/company interaction emails and evaluating multiple classifiers. Expert Systems with Applications 36 (3), 6127–6134.

Culp, M., Johnson, K., Michailidis, G., 2012. R package ada: an r package for stochastic boosting.

Datta, P., Masand, B., Mani, D. R., Li, B., Dec. 2000. Automated cellular modeling and prediction on a large scale. Artificial Intelligence Review 14 (6), 485–502.

De Bock, K. W., Van den Poel, D., 2011. An empirical evaluation of rotation-based ensemble classifiers for customer churn prediction. Expert Systems with Applications 38 (10), 12293–12301.

De Bock, K. W., Van den Poel, D., Jun. 2012. Reconciling performance and interpretability in customer churn prediction using ensemble learning based on generalized additive models. Expert Systems with Applications 39 (8), 6816–6826.

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

Dietterich, T. G., 1998. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10 (7), 1895–1923.

Dreiseitl, S., Ohno-Machado, L., Dec. 2002. Logistic regression and artificial neural network classification models: a methodology review. Journal of Biomedical Informatics 35 (5-6), 352–359.

Eiben, A. E., Koudijs, A. E., Slisser, F., 1998. Genetic modelling of customer retention. Genetic Programming. First European Workshop, EuroGP'98. Proceedings, 178–86.

Ellison, N. B., Steinfield, C., Lampe, C., Jul. 2007. The benefits of facebook "friends": Social capital and college students' use of online social network sites. Journal of Computer-Mediated Communication 12 (4), 1.

Facebook, Sep. 2013a. News feed FYI: a window into news feed.
URL `https://www.facebook.com/business/news/News-Feed-FYI-A-Window-Into-News-Feed/`

Facebook, Sep. 2013b. News feed FYI: more relevant ads in news feed | facebook newsroom.
URL `http://newsroom.fb.com/news/2013/09/news-feed-fyi-more-relevant-ads-in-news-feed/`

Facebook, 2014. Newsroom - key facts.
URL `http://newsroom.fb.com/Key-Facts`

Freund, Y., Schapire, R., 1996. Experiments with a new boosting algorithm. In: Machine Learning. Proceedings of the Thirteenth International Conference (ICML '96). Bari, Italy, pp. 148–156.

Friedman, J., Hastie, T., Tibshirani, R., Feb. 2010. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33 (1), 1–22.

Friedman, J., Hastie, T., Tibshirani, R., 2013. R package glmnet: Lasso and elastic-net regularized generalized linear models.

Friedman, J. H., Oct. 2001. Greedy function approximation: A gradient boosting machine. Annals of Statistics 29 (5), 1189–1232.

Friedman, J. H., Feb. 2002. Stochastic gradient boosting. Computational Statistics & Data Analysis 38 (4), 367–378.

Friedman, J. H., Meulman, J. J., May 2003. Multiple additive regression trees with application in epidemiology. Statistics in Medicine 22 (9), 1365–1381.

Glady, N., Baesens, B., Croux, C., 2009. Modeling churn using customer lifetime value. European Journal of Operational Research 197 (1), 402–411.

Gómez, D., Figueira, J. R., Eusébio, A., Apr. 2013. Modeling centrality measures in social network analysis using bi-criteria network flow optimization problems. European Journal of Operational Research 226 (2), 354–365.

Guisan, A., Edwards, T. C., Hastie, T., Nov. 2002. Generalized linear and generalized additive models in studies of species distributions: setting the scene. Ecological Modelling 157 (2-3), 89–100.

Hand, D. J., Sep. 2005. Good practice in retail credit scorecard assessment. Journal of the Operational Research Society 56 (9), 1109–1117.

Hanley, J., Mcneil, B., 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology 143 (1), 29–36.

Hartmann, W. R., Manchanda, P., Nair, H., Bothner, M., Dodds, P., Godes, D., Hosanagar, K., Tucker, C. E., Dec. 2008. Modeling social interactions: Identification, empirical methods and policy implications. Marketing Letters 19 (3-4), 287–304.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning - Data Mining, Inference, and Prediction, second edition Edition. Springer Series in Statistics. Springer.

Hellmann, T., Staudigl, M., May 2014. Evolution of social networks. European Journal of Operational Research 234 (3), 583–596.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., 2010. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan.

Hung, S.-Y., Yen, D. C., Wang, H.-Y., Oct. 2006. Applying data mining to telecom churn management. Expert Systems with Applications 31 (3), 515–524.

Hwang, H., Jung, T., Suh, E., Feb. 2004. An LTV model and customer segmentation based on customer value: a case study on the wireless telecommunication industry. Expert Systems with Applications 26 (2), 181–188.

Janitza, S., Strobl, C., Boulesteix, A.-L., Apr. 2013. An AUC-based permutation variable importance measure for random forests. BMC Bioinformatics 14 (1), 119.

Kim, Y., 2006. Toward a successful CRM: variable selection, sampling, and ensemble. Decision Support Systems 41 (2), 542–553.

King, R., Feng, C., Sutherland, A., 1995. Statlog - comparison of classification algorithms on large real-world problems. Applied Artificial Intelligence 9 (3), 289–333.

Kumar, D. A., Ravi, V., 2008. Predicting credit card customer churn in banks using data mining. International Journal of Data Analysis Techniques and Strategies 1 (1), 4–28.

Lampe, C., Ellison, N., Steinfield, C., 2007. A Familiar Face(book): Profile Elements as Signals in an Online Social Network.

Larivière, B., Van den Poel, D., 2005. Predicting customer retention and profitability by using random forests and regression forests techniques. Expert Systems with Applications 29 (2), 472–484.

Lemmens, A., Croux, C., May 2006. Bagging and boosting classification trees to predict churn. Journal of Marketing Research (JMR) 43 (2), 276–286.

Lemon, K. N., White, T. B., Winer, R. S., Jan. 2002. Dynamic customer relationship management: Incorporating future considerations into the service retention decision. Journal of Marketing 66 (1), 1–14.

Lewis, K., Kaufman, J., Gonzalez, M., Wimmer, A., Christakis, N., Oct. 2008. Tastes, ties, and time: A new social network dataset using facebook.com. Social Networks 30 (4), 330–342.

Liaw, A., Wiener, M., 2002. Classification and regression by randomForest. R News 2 (3), 18–22.

Liaw, A., Wiener, M., 2012. R package randomForest: breiman and cutler's random forests for classification and regression.

Lima, E., Mues, C., Baesens, B., Aug. 2009. Domain knowledge integration in data mining using decision tables: case studies in churn prediction. Journal of the Operational Research Society 60 (8), 1096–1106.

Madden, G., Savage, S. J., Coble-Neal, G., Jul. 1999. Subscriber churn in the australian ISP market. Information Economics and Policy 11 (2), 195–207.

Martin-Barragan, B., Lillo, R., Romo, J., Jan. 2014. Interpretable support vector machines for functional data. European Journal of Operational Research 232 (1), 146–155.

Matignon, R., Aug. 2005. Neural Network Modeling Using Sas Enterprise Miner. AuthorHouse.

McCulloch, W., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., 2012. R package e1071: Misc functions of the department of statistics (e1071),.

Migueis, V., Van den Poel, D., Camanho, A., Falcao e Cunha, J., 2012a. Modeling partial customer churn: On the value of first product-category purchase sequences. Expert Systems with Applications 39 (12), 11250–11256.

Migueis, V. L., Van den Poel, D., Camanho, A. S., Cunha, J. F. e., Dec. 2012b. Predicting partial customer churn using markov for discrimination for modeling first purchase sequences. Advances in Data Analysis and Classification 6 (4), 337–353.

Mozer, M., Wolniewicz, R., Grimes, D., Johnson, E., Kaushansky, H., 2000. Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry. IEEE Transactions on Neural Networks 11 (3), 690 –696.

Nemenyi, P., 1963. Distribution-free multiple comparisons. Ph.D. thesis, Princeton University.

Neslin, S. A., Gupta, S., Kamakura, W., Junxiang Lu, Mason, C. H., May 2006. Defection detection: Measuring and understanding the predictive accuracy of customer churn models. Journal of Marketing Research (JMR) 43 (2), 204–211.

Ngai, E., Xiu, L., Chau, D., 2009. Application of data mining techniques in customer relationship management: A literature review and classification. Expert Systems with Applications 36 (2), 2592–2602.

Platt, J., 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. Advances in Large Margin Classifiers. MIT Press, Cambridge, MA.

Prinzie, A., Van den Poel, D., 2006. Incorporating sequential information into traditional classification models by using an element/position-sensitive SAM. Decision Support Systems 42 (2), 508–526.

Prinzie, A., Van den Poel, D., Apr. 2008. Random forests for multiclass classification: Random MultiNomial logit. Expert Systems with Applications 34 (3), 1721–1732.

Prinzie, A., Van den Poel, D., Jun. 2011. Modeling complex longitudinal consumer behavior with dynamic bayesian networks: an acquisition pattern analysis application. Journal of Intelligent Information Systems 36 (3), 283–304.

Provost, F., Fawcett, T., Kohavi, R., 1998. The case against accuracy estimation for comparing induction algorithms. In: Shavlik, J. (Ed.), Machine Learning. Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98). Morgan Kaufmann Publishers, Madison, WI, USA, pp. 445–453.

R Core Team, R., 2013. R package stats: R statistical functions.

Ripley, B., 1996. Pattern Recognition and Neural Networks. Cambridge University Press.

Ripley, B., 2013. R package nnet: Feed-forward neural networks and multinomial log-linear models.

Rossi, P. E., McCulloch, R. E., Allenby, G. M., 1996. The value of purchase history data in target marketing. Marketing Science 15 (4), 321–340.

Smith, K. A., Willis, R. J., Brooks, M., 2000. An analysis of customer retention and insurance claim patterns using data mining: a case study. Journal of the Operational Research Society 51 (5), 532–541.

Spackman, K. A., 1991. Maximum likelihood training of connectionist models: comparison with least squares back-propagation and logistic regression. In: Proceedings of the Annual Symposium on Computer Application in Medical Care. pp. 285–289.

Thorleuchter, D., Van den Poel, D., Dec. 2012. Predicting e-commerce company success by mining the text of its publicly-accessible website. Expert Systems with Applications 39 (17), 13026–13034.

Thorleuchter, D., Van den Poel, D., Prinzie, A., Feb. 2012. Analyzing existing customers' websites to improve the customer acquisition process as well as the profitability prediction in b-to-b marketing. Expert Systems with Applications 39 (3), 2597–2605.

Van den Poel, D., Buckinx, W., Oct. 2005. Predicting online-purchasing behaviour. European Journal of Operational Research 166 (2), 557–575.

Van den Poel, D., Lariviere, B., 2004. Customer attrition analysis for financial services using proportional hazard models. European Journal of Operational Research 157 (1), 196–217.

Venables, W. N., Ripley, B. D., 2002. Modern Applied Statistics with S, 4th Edition. Springer, New York.

Venkatesh, K., Ravi, V., Prinzie, A., Poel, D. V. d., Jan. 2014. Cash demand forecasting in ATMs by clustering and neural networks. European Journal of Operational Research 232 (2), 383–392.

Verbeke, W., Martens, D., Mues, C., Baesens, B., Mar. 2011. Building comprehensible customer churn prediction models with advanced rule induction techniques. Expert Systems with Applications 38 (3), 2354–2364.

Verhaert, G. A., Van den Poel, D., Feb. 2011. Improving campaign success rate by tailoring donation requests along the donor lifecycle. Journal of Interactive Marketing 25 (1), 51–63.

Vert, J.-P., Tsuda, K., Schölkopf, B., 2004. A primer on kernel methods. In: Schölkopf, B., Tsuda, K., Vert, J.-P. (Eds.), Kernel Methods in Computational Biology. Computational Molecular Biology. MIT Press, pp. 35–70.

Weerahandi, S., Moitra, S., 1995. Using survey data to predict adoption and switching for services. Journal of Marketing Research 32 (1), 85–96.

Wei, C.-P., Chiu, I.-T., 2002. Turning telecommunications call details to churn prediction: a data mining approach. Expert Systems with Applications 23 (2), 103–112.

Xie, Y., Li, X., Ngai, E., Ying, W., 2009. Customer churn prediction using improved balanced random forests. Expert Systems with Applications 36 (3), 5445–5449.

4

Hybrid Ensembles: Many Ensembles is Better Than One

Ballings, M, Vercamer, D., Van den Poel, D. Hybrid Ensembles: Many Ensembles is Better Than One. Submitted for peer review in 2013 to Pattern Recognition.

4.1 Abstract

The purpose of this paper is to assess the added value of algorithm- induced diversity over and above data- induced diversity in ensemble design. We develop a Hybrid Ensemble consisting of six sub-ensembles: Bagged Logistic Regression, Random Forest, Kernel Factory, Bagged Support Vector Machines, Stochastic Boosting, and Bagged Neural Networks. We test the algorithm on eleven data sets using five times twofold cross-validation. The Hybrid Ensemble significantly and consistently outperforms the Single Best sub-ensemble on all data sets when the authority method or Self Organizing Migrating Algorithm is used for weight estimation. Analyses also indicate that the Hybrid Ensemble yields increasingly important classification improvements with increasingly difficult tasks. To the best of our knowledge this study is the first to assess the added value of algorithm-induced diversity over and above data-induced diversity in ensemble design. To make our results replicable we submitted an open- source R-software package called *hybridEnsemble* to CRAN.

4.2 Introduction

Theoretical and experimental research has shown that diversity is of chief importance in classifier ensemble design (Krogh and Vedelsby, 1995; Kuncheva and Whitaker, 2003; Kuncheva, 2004; Brown et al., 2005). Given an adequate combination rule and sufficient diversity, even poorly performing team members (i.e., base classifiers) can engender a highly accurate ensemble (Kuncheva and Whitaker, 2003). Although accurate base classifiers are still favorable (Krogh and Vedelsby, 1995), it is especially important that their errors are uncorrelated (i.e., diverse). Research has proven that higher diversity results in a lower generalization error (Opitz and Shavlik, 1996). The result is that ensembles typically outperform the Single Best classifier. While in theory, one could optimize explicitly for diversity (Kuncheva and Whitaker, 2003), no successful attempts have been made up to this point (Tang et al., 2006; Gal-Or et al., 2005). Therefore literature has identified two main strategies to implicitly create diversity: (1) data perturbation and (2) algorithm variety. The most followed designs adhere to the data- strategy. Bootstrap aggregating (Bagging) (Breiman, 1996) was the first method to effectively employ the data perturbation strategy. Similarly, Boosting (Freund and Schapire, 1996) and Random Subspaces (Ho, 1998) also showed the merits of applying this approach. Random Forest, introduced by Breiman (2001), has been the most influential work in this area as it combined both Bagging and Random Subspaces to further increase accuracy.

While data perturbation has evolved to become the most popular diversity generation strategy, algorithm variation strategies have been lingering in the background. Examples of algorithm variation strategies are varying parameters and combining different algorithms. One of the reasons may be that researchers do not want to invest time into the intricacies of tuning a large number of different classification algorithms (King et al., 1995) while the data- strategy is much simpler to implement. Another reason might be that scholars want to focus on weak base classifiers (stumps) in order to more easily find significant benefits through ensembling (Ko et al., 2008). Ensembles of Neural Networks have often relied on differences in parameter initialization (Partridge and Yates, 1996; Yates and Partridge, 1996) but several experiments revealed that this is the least effective way to create diversity (Brown et al., 2005). Still, competitive results can also be achieved through the variation of algorithms. Michalski et al. (1994); Woods et al. (1997); Mitchell (1997) already showed that the difference in inductive bias could be beneficial toward increasing accuracy. In more recent work, Menahem et al. (2009) combined C4.5 Decision Tree, Naive Bayes, KNN, VFI and OneR for improving malware detection and Zhang (2011) used Random forest, Neural Networks and Support Vector Machines to create an ensemble that worked well on biomedicine data. Another study by van der Laan et al. (2007) used seven different base classifiers to create a Super Learner for regression. However, both data- and algorithm- strategies have mostly been studied in isolation and no study has explicitly attempted to combine them. As more diversity should be beneficial to the combined predictive performance, a combination of the two strategies could lead to higher ensemble accuracy.

In response to this gap we will assess the added value of algorithm-driven diversity to ensemble performance over and above data-driven diversity while automatically tuning the classifier parameters. More specifically, we will investigate whether including different algorithms in an ensemble can outperform top performing ensembles such as Random Forest and Boosting that make use of data- strategies. To assess this strategy, we employ a select number of popular and high performing learning methods as base classifiers: Random Forest (Breiman, 2001), Stochastic Boosting (Friedman, 2002), Artificial Neural Networks (McCulloch and Pitts, 1943), Logistic Regression (Berkson, 1944), Support Vector Machines (Cortes and Vapnik, 1995) and Kernel Factory (Ballings and Van den Poel, 2013a).

The rest of this paper is organized as follows: in Section 4.3 we will give an overview of related literature. Section 4.4 describes our Hybrid Ensemble. The design of experiments is provided in Section 4.5 with results in Section 4.6. Finally in Section 4.7 we conclude this paper and discuss limitations and directions for future research.

4.3 Related Work

The concept of diversity is one of the most popular explanations for the success of ensemble methods (Kuncheva and Whitaker, 2003). As a decrease in the ensemble error relates to some form of diversity, understanding it is essential (Zhou, 2012). In a regression setting, bias-variance-covariance (BVC) decomposition (Geman et al., 1992) and ambiguity decomposition (Krogh and Vedelsby, 1995) have been used as an effective framework for understanding diversity. Despite this widespread attention, in classification a clear framework and definition are yet to be formulated (Tang et al., 2006; Kuncheva, 2003). A promising attempt by Didaci et al. (2013) includes an ambiguity-like and BVC-like decomposition for classification that requires further investigation. Moreover, techniques such as Bagging effectively show the merits of using a diverse set of classifiers but the underlying objective function that is implicitly optimized by these techniques is still under investigation (Didaci et al., 2013). Still it is widely believed that diversity is a quantifiable property of ensembles that can be used in their design. Theoretical analysis (Kittler et al., 1998; Brown et al., 2005) supports this hypothesis as well as the numerous experiments with diverse classifiers that have been undertaken.

Diversity creation, part of what is often called coverage optimization (Ho, 2002), has been categorized in different ways throughout the years. Typically at a higher level the distinction is made between explicit and implicit diversity creation (Brown et al., 2005; Tang et al., 2006). In explicit diversity creation a specific measure is used and optimized to increase the error diversity among the base classifiers. However, most measures that were proposed by Kuncheva and Whitaker (2003) have been proven to be ineffective (Tang et al., 2006) as the relationship between an improved diversity measure and the ensemble accuracy is ambiguous. It also raised the question whether ensemble accuracy itself should be preferred to using an estimate of diversity (Ruta and Gabrys, 2005; Didaci et al., 2013). Brown and Kuncheva (2010) argued that there may be such a thing as ”good” and ”bad” diversity. The ultimate goal should be to increase the amount of ”good” diversity and reduce ”bad” diversity. In this case, ”good” diversity is defined as the disagreement among the base classifiers where the ensemble is right. ”Bad” diversity is the disagreement among the base classifiers where the ensemble is wrong. Still, no explicit measure has been found that can be used for ensemble design (Didaci et al., 2013). While explicit diversity creation has mostly failed to consistently obtain high accuracies, implicit diversity has done much better. As mentioned in the introduction, implicit diversity relies on some form of randomness (initial weights, data perturbation, used algorithms) (Brown et al., 2005) to create different regions in the search space.

Diving deeper into the realm of implicit diversity generation, the most effective ways to create diverse and accurate classifiers is by means of implicit heuristic techniques. There are two broad methods that encompass extant literature: *data-induced* and *algorithm-induced* diversity creation. Data-induced diversity has been called different things by different authors. Sharkey (1999), Brown et al. (2005) and Rokach (2010) called it the manipulation or use of training data. Baumgartner and Serpen (2012) and Zhang and Zhou (2011) called it ensembles with data perturbation. Rokach (2010) also included the change of the target attribute representation and the partitioning of the search space. Popular examples of data-driven methods include Bagging (Breiman, 1996) which creates diversity through the instances, Random Subspaces (Ho, 1998; Bryll et al., 2003) that focuses on the features and Random Forest (Breiman, 2001) which combines both. Analogously, algorithm-induced diversity also exists in multiple forms. For Sharkey (1999) this would be the architecture of the networks, the parameter settings of the inducer and the inducer itself. In the taxonomy of Brown et al. (2005) this is the starting point in the hypothesis space and its traversal as well as the manipulation of the architecture part of the set of accessible hypotheses. Rokach (2010) covers it with the manipulation of the inducer and hybridized methods. Lastly, Baumgartner and Serpen (2012) and Zhang and Zhou (2011) called it homogeneous ensembles with different parameters and heterogeneous ensembles. In algorithm-driven diversity, most attention has been on designs with different weights or parameters (Ueda, 2000; Windeatt, 2005). Unfortunately authors recognized it is the least effective way to create diversity (Brown et al., 2005). Michalski et al. (1994) and Tsoumakas et al. (2005) found that multi-strategy ensembles are more effective. In this paper, data-induced diversity focuses on the data perturbation methods while algorithm-induced diversity relies on multiple types of inducers, also known as multi-strategy or Hybrid Ensembles. Both diversity generation schemes have been effectively used to obtain high ensemble accuracy.

While data perturbation techniques have flourished in recent years, the use of algorithm variation has remained largely unexplored. Although several papers recognized that combining multiple inductive biases is an effective way to create diversity, only few scholars have used this strategy so far. Furthermore, even less authors recognized that the integration of both schemes could be beneficial in improving the ensemble accuracy. Table 4.1 summarizes the literature by including a set of ground-breaking, pivotal papers that use data variation and algorithm variation in classification.

Table 4.1: Hybrid Diversity Generation Strategies

Data set	Data	Algorithm	Algorithm → Data	Data → Algorithm
Breiman (1996)	x			
Freund and Schapire (1996)	x			
Ho (1998)	x			
Breiman (2001)	x			
Michalski et al. (1994)		x		
Woods et al. (1997)		x		
Wang et al. (2000)		x		
Tsoumakas et al. (2005)		x		
Canuto et al. (2007)		x		
Menahem et al. (2009)		x		
Nascimento and Coelho (2009)	x	x	x	
This study	x	x		x

One example of integration of both strategies can be found in Nascimento and Coelho (2009) where multiple inducers are integrated into a Bosting scheme. However, as Table 4.1 shows, hybrid forms of ensembles where data-induced ensembles are combined have not yet been explored. In regression, a first attempt was made by Yu et al. (2007) but only three base classifiers were used and no extension was made toward classification. As the effectiveness of both strategies in ensemble classification has only been proven in isolation, this study could deliver insightful results. This paper aims to investigate whether algorithm-induced diversity significantly adds value over and above data-induced diversity in a classification setting.

4.4 Hybrid Ensemble

In this study we include different sub-ensembles that employ a data- strategy into a Hybrid Ensemble. By comparing the Hybrid Ensemble's classification performance with the Single Best sub-ensemble we are able to assess the added value of algorithm-induced diversity over data-induced diversity. In the following subsection we will justify our choice of those sub-ensembles.

4.4.1 Base classifiers

The industrially preferred (Ruta and Gabrys, 2005) classifier selection method is called Single Best: ensembles that have performed well in the past are evaluated, and the best performing one is selected. Our proposed Hybrid Ensemble will only be relevant if it outperforms the industrially preferred method. In this light, we have searched extant literature and selected six algorithms, new and old, that perform well on a multitude of data sets: Logistic Regression (Berkson, 1944), Artificial Neural Networks (McCulloch and Pitts, 1943), Support Vector Machines (Cortes and Vapnik, 1995), Random Forest (Breiman, 2001), Stochastic Boosting (Friedman, 2001), and Kernel Factory (Ballings and Van den Poel, 2013a).

As mentioned above, to be able to assess the added value of algorithm variation over data variation, we need to ensure that all ensemble members are ensembles themselves employing a data- strategy. Random Forest, Stochastic Boosting and Kernel Factory inherently use a data-strategy. For the other three algorithms (Logistic Regression, Neural Networks and Support Vector Machines) we use Bagging to make the ensembles. We choose Bagging because it is one of the most effective data- strategies, easy to implement, and inherently parallel. Bagging consistently outperforms single classifiers in case of Neural Networks (Zhou et al., 2002), Logistic Regression (Kim, 2006), and Support Vector Machines (Valentini et al., 2003). The next step in the process will be to combine these members in a Hybrid Ensemble (see Section 4.4.2).

The selection of these six algorithms is based on a number of studies that rigorously compare multiple algorithms (up to thirty-three learning methods on up to thirty-six data sets). King et al. (1995) compare seventeen methods on twelve real-life data sets ranging from image analysis, medicine, and engineering to finance. Cooper et al. (1997) evaluate eight models on a medical data set. Bauer and Kohavi (1999) investigate multiple methods on fourteen data sets including credit scoring, image analysis, and letter recognition. Lim et al. (2000) benchmark thirty-three old and new classification algorithms on a diverse set of sixteen data sets (and the sixteen same data sets but with added noise). Breiman (2001) compares two algorithms on 19 data sets including problems of credit scoring, medical and engineering problems. Perlich et al. (2004) uses two algorithms to analyze thirty-six different data sets about a multitude of problems (e.g., pricing, bacteria detection, product evaluation, game outcomes, and credit scoring). Caruana and Niculescu-Mizil (2006) provide an empirical comparison of ten learning methods on eleven data sets. Neslin et al. (2006) discuss results of a benchmark considering seven methods in a marketing context. Coussement and Van den Poel (2008) evaluate three methods in an operational marketing context. Burez and Van den Poel (2009) compare four algorithms on six marketing data sets. Finally, Ballings and Van den Poel (2013a) assess three learning methods on fourteen data sets including, among others, medical and financial data.

All these benchmarks denote that the selected classifiers in this study are top performers. One could argue that including more methods could benefit ensemble performance. There are four reasons not to increase the ensemble size. First, including more team members would increase learning times and computational effort for both the base classifiers (Margineantu and Dietterich, 1997) and the combination methods to an impractical level. Second, small ensembles have, in many cases, near-optimal performance (Margineantu and Dietterich, 1997). Third, diversity has a strong effect in the ensemble performance when using less than ten classifiers (Kuncheva, 2004; Canuto et al., 2007). Fourth, we included only the top performing ensembles found in extant literature, given that the purpose of this paper is to improve upon the industrially preferred Single Best method.

Whether an ensemble of the aforementioned classifiers can outperform the Single Best depends on the diversity of the classifiers' predictions (Ruta and Gabrys, 2005). Some of the above studies distill which data characteristics will lead those classifiers to perform differently (result in different predictions). Logistic Regression performs worse than classification trees if data are not normally distributed and if there are many categorical predictors (King et al., 1995), but performs better on smaller data sets (Perlich et al., 2004). Trees perform better than Logistic Regression on data with extreme distributions (King et al., 1995) and data sets with a higher signal to noise (Perlich et al., 2004). While Logistic Regression captures the covariates linearly (Dreiseitl and Ohno-Machado, 2002), decision trees are inherently non-linear. Random Forest (trees) can handle high variance situations (Bauer and Kohavi, 1999) and Boosting reduces both bias and variance (Bauer and Kohavi, 1999). Support Vector Machines are very sensitive to the choice of the kernel function and will either work very well, or very poorly depending on the data (and consequently the right kernel) (Ballings and Van den Poel, 2013a). This limitation is similar for Kernel Factory, however, it is capable of capturing remaining non-linearities in the kernel matrix (Ballings and Van den Poel, 2013a). Neural Networks are universal approximators. While the functional forms of Logistic Regression and Neural Networks differ, both algorithms would be identical if Neural Networks would not have a hidden layer and if the logistic activation function is used (Dreiseitl and Ohno-Machado, 2002). The hidden layer allows Neural Networks to model non-linear functions of the inputs. Nevertheless, this extra flexibility requires more parameter tuning, hence depending on the data it is well possible that the simpler Logistic Regression outperforms a Neural Network. One could also argue that these techniques will most likely result in diverse predictions by investigating their inner workings. As displayed in Table 4.2 there is a large diversity in the main features, objective functions and optimization methods of the six base classifiers used in this study.

We hypothesize that algorithm-induced diversity will most likely lead to diverse outputs and subsequently to increased predictive performance of the Hybrid Ensemble over the Single Best. However we do not expect to find very big differences because all the base classifiers are homogeneous ensembles that already employ a diversity strategy (through data) and hence are considerably strong. Whether the ensemble of ensembles that we propose in this study will be able to outperform the Single Best ensemble depends on how much diversity can be induced.

To summarize, our Hybrid Ensemble consists of data-driven inducers that each use a different learning method. By combining these base classifiers into one ensemble, we create a meta-ensemble that is also algorithm-driven.

4.4.2 Classifier combination

While the focus of this paper is on diversity generation mechanisms, an essential part of ensemble design is also its combination rule. In literature, different fusion methods exist depending

Table 4.2: Characteristics of the base classifiers

Algorithm	Main features	Objective function	Optimization method
Ensemble of Neural Networks	Identical to Logistic Regression if no hidden layer and if logistic activation function is used. Semi/non parametric. Hidden layer(s). Bagging.	Conditional log loss	Quasi Newton Method
Ensemble of Support Vector Machines	Maximum margin seperating hyperplane and soft margin. Kernels. Bagging.	Hinge loss	Constrained quadratic programming
Ensemble of Logistic Regressions with Lasso	Linear combination of features. Logistic link function. Bagging.	Penalized log loss	Newton method
Random Forest	Parallel- independently built trees. Random feature selection. Bagging.	No explicit global loss function. Locally: entropy=log loss	Greedy search
Kernel Factory	Kernels. Base classifiers are Random Forests. Random Forest handles remaining non-linearities from kernels.	No explicit global loss function. Locally: entropy=log loss	Greedy search
Stochastic AdaBoost	Iterative- dependently built trees. Focus on poorly classified instances. Hybrid bagging and boosting.	Exponential loss	Gradient descent

on the output that is received from the base classifiers. While some form of (weighted) voting has become the most frequently used method in classifier fusion, it is not necessarily the best option. A lot of information is lost as it only uses class label output. Confidences or *a posteriori* probabilities contain the highest amount of information (Xu et al., 1992) and are able to reduce the generalization error (Bauer and Kohavi, 1999). Therefore in our hybrid design we calibrate all of our models to ensure we have measurement level output for every base classifier. As a result, we can fuse our classifiers using weighted averaging. In this study we restrict ourselves to linear combinations because we want to limit the number of parameters that needs to be estimated in order to keep the analysis tractable.

To calculate the weights, one can make use of fixed rules or trained weights (Roli et al., 2002). While fixed rules have a very small time complexity and provide simplicity, their result is expected to be worse than that of the trained ones. The easiest way to combine the different measures is to take the simple average. The performance of this simple method is often close to that of the weighted average but in most cases, a weighted average is able to outperform the simple average (Fumera and Roli, 2005). In performance- or authority- based weighting the weight of each classifier is set proportional to its performance on a validation set (Opitz and Shavlik, 1996). This method typically performs better albeit at the cost of more computational effort.

Weights can be trained by either a statistical method or a general purpose solver. The latter category has the advantage that the objective function can be chosen freely to fit the application (in this case AUC) (Elkan, 2013) and that it is more likely to find global optima for the parameters (Myung, 2003). Although not the primary focus of this study, we tried to benchmark as many methods as possible in each category.

For the statistical methods we use Non-Negative Binomial Likelihood (Sra et al., 2008), Goldfarb-Idnani Non-Negative Least Squares (Goldfarb and Idnani, 1983), and Lawson-Hanson Non-Negative Least Squares (Lawson and Hanson, 1987). These methods have the advantage that they enforce the non-negativity constraint, as such preserving the inherent characteristics of the weighted solution that we seek (a weighting value between zero and one) (Chen and Plemmons, 2007).

In the category of general purpose solvers we included all the following population- based methods and single solution based methods. Within the population- based methods, Genetic Algorithms (Holland, 1975) are the most popular. They have already been extensively applied in ensemble fusion and selection (Kim et al., 2003; Cho, 1999; Wu et al., 2001). Recently, Differential Evolution (Storn and Price, 1997) has shown promising results for simultaneous selection and fusion (de Lima et al., 2012) as well. A last population- based method that has shown good results is Particle Swarm Optimization (PSO). It is a popular form of Swarm Intelligence. It is based on the coordinated way that flocks of birds and schools of fish look for food.

In the single solution based methods, Generalized Simulated Annealing seems to be the most popular method. Introduced by Tsallis and Stariolo (1996), it is a form of stochastic gradient descent. In recent years, other methods such as Memetic Algorithm (Molina et al., 2010), Self-Organizing Migrating Algorithm (Molina et al., 2010) and Tabu Search Algorithm (Glover, 1977, 1986, 1989, 1990b,a) have also been used for continuous parameter optimization. Although they present very good results on other classes of problems, they have not been applied to weight estimation for classifier fusion.

In sum, we use three statistical methods, three population- based general purpose methods, four single solution based methods, the authority- based weighting method, and the simple mean. We benchmark these methods with the Single Best.

4.4.3 Pseudo code

Algorithm 4 and 5 are respectively the pseudo-code of the Hybrid Ensemble's training and prediction phase. Let LR be Bagged Logistic Regression, let RF be Random Forest, let AB be AdaBoost, let KF be Kernel Factory, let NN be Bagged Neural Networks and, let SV be Bagged Support Vector Machines. Furthermore, let GA be Genetic Algorithm, let DEA be Differential Evolutionary Algorithm, let GSA be Generalized Simulated Annealing, let MALSC be Memetic Algorithm with Local Search Chains, let PSO be Particle Swarm Optimization, let SOMA be Self-Organising Migrating Algorithm, let TSA be Tabu Search Algorithm, let NNBL be Non-negative binomial likelihood, let GINNLS be Goldfarb-Idnani Non-negative least squares, let LHNNLS be Lawson-Hanson Non-negative least squares, let AUTH be authority- based weighting and let MEAN be the simple mean.

An important step in the algorithm is calibration of the members' classifier outputs. This step is required for a meaningful combination of the probabilistic outputs of the base classifiers. Calibration consists in transforming the raw output scores to probabilities. A classifier is well calibrated when the empirical class membership $P(c|p(x) = t)$, with c the observed class, $p(x)$ the predicted probability, and t the threshold value, converges to $p(x) = t$ when the number of classified instances goes to infinity (Murphy and Winkler, 1977). More intuitively, if we consider the instances to which a classifier assigns a probability $p(x) = 0.8$, then 80% of these instances should be members of the observed class c (Zadrozny and Elkan, 2002).

For each classifier, the calibration algorithm first bins the classifier scores (Coussement and Buckinx, 2011). The number of bins is determined by cross- validation and the bins are of equal size. Next, for all bins two quantities are computed: the mean score P_{raw} and the proportion of events in the response vector P_{cal}. The latter can be conceived of as an approximation of the real probability. The final step consists in learning $P_{cal} = f(P_{raw})$. The function f is learned by a regression forest (Breiman, 2001) and is applied to unseen data in the prediction phase.

Base classifier parameter tuning is performed by cross-validation using x_{train}, y_{train}, $x_{validate}$, and $y_{validate}$. The combiners that are tuned use $\hat{y}_{validate}$ and $y_{validate}$. Details about tuning are provided in Sections 4.5.1.1 and 4.5.1.2.

```
Input:
• $x$=predictor variables
• $y$=response variable with class labels $\{0,1\}$
• combine= one of the following combination methods {GA, DEA,
  GSA, MALSC, PSO, SOMA, TSA, NNBL, GINNLS, LHNNLS,
  AUTH, MEAN }
• member parameters=parameters of base classifiers (see 4.5.1.1)
• combination parameters=parameters of combiners (see 4.5.1.2)

Classifier Generation:
Randomly divide $x$ into $x_{train}$ (50% of instances) and $x_{validate}$ (50%)
Make the same split for $y$: $y_{train}$ and $y_{validate}$
Algorithms ← (LR, RF, AB, KF, NN, SV)
for Algorithms do
    Tuned Parameters ← tune($x_{train}$, $y_{train}$, $x_{validate}$, $y_{validate}$)
    Classifiers ← train($x_{train}$, $y_{train}$, Tuned Parameters)
    $\hat{y}_{validate}$ ← predict(Classifiers, $x_{validate}$)
    Calibrators ← train calibrator($\hat{y}_{validate}$, $y_{validate}$)
    $\hat{y}_{validate}$ ← calibrate(Calibrators, $\hat{y}_{validate}$)
    Evaluations ← evaluate($\hat{y}_{validate}$, $y_{validate}$)
    Classifiers ← train($x$,$y$, Tuned Parameters)
end

Classifier combination:
if combine one of {GA, DEA, GSA, MALSC, PSO, SOMA, TSA, NNBL, GONNLS, LHNNLS}
then
    weights ← optimize classifier weights ($\hat{y}_{validate}$, $y_{validate}$)
else if combine == AUTH then
    weights ← evaluations/sum(evaluations)
else if combine == MEAN then
    weights ← (1/6,1/6,1/6,1/6,1/6,1/6)

Result: Calibrators, Classifiers, Weights
```

algorithm 4: Pseudo code training phase of Hybrid Ensemble

```
Input:
• $object$=trained Hybrid Ensemble (Calibrators, Classifiers, Weights)
• $x$=new predictor variables

Predict using all Classifiers:
for all Classifiers do
    $\hat{y} \leftarrow$ predict(Classifiers, $x$)
    $\hat{y} \leftarrow$ calibrate(Calibrators, $\hat{y}$)
end
Compute weighted average: $\hat{Y} \leftarrow \langle weights, \hat{y} \rangle$
Result: $\hat{Y}$
```

algorithm 5: Pseudo code prediction phase of Hybrid Ensemble

4.5 Experimental Design

4.5.1 Implementation details

All analyses are performed using *R* (R Core Team, 2013) version 3.0.2. We developed an open-source package of our algorithm called *hybridEnsemble* and submitted it to CRAN.

4.5.1.1 Base algorithm parameters

This section covers our choices for the base classifiers' parameters. Because not all algorithms can handle categorical variables, all categorical predictors are first transformed to dummy variables {0,1} before any modeling takes place. For Bagged Logistic Regression, Bagged Support Vector Machines and Bagged Neural Networks we opt for ten members to keep learning times and computational effort (Margineantu and Dietterich, 1997) in a feasible range. Small ensembles have been proven to have near-optimal performance (Margineantu and Dietterich, 1997) and diversity only has a strong effect when using up to ten classifiers (Kuncheva, 2004; Canuto et al., 2007). We build the members on bootstrap samples with size equal to the original sample and we aggregate the predictions with the simple mean.

To avoid overfitting, we use the lasso approach to Regularized Logistic Regression. Lasso imposes a bound on the sum of the absolute values of the coefficients. Coefficients are shrunk towards zero (Guisan et al., 2002). The shrinkage parameter is determined by cross-validation. We use the *glmnet* R package (Friedman et al., 2010, 2013) to fit the model. The α parameter is set to one to obtain the lasso method and we compute the sequence of 100 λ's by setting *nlambda*.

Random Forest requires only to set the number of variables to evaluate at each split and the number of trees in the ensemble. We follow Breiman's recommendation (Breiman, 2001) and set the number of variables to the square root of the total number of variables and use a large number of trees (500). We use the *randomForest* R package (Liaw and Wiener, 2012, 2002).

Stochastic Boosting improves on the original deterministic Boosting algorithms (Freund and Schapire, 1996) by incorporating randomness as an integral part of the procedure (Friedman, 2002). The number of terminal nodes in the base classifiers and the number of iterations are two important parameters. We set the maximum number of nodes to eight by setting the maximum depth of the trees to three which is in line with the Friedman's (2002) advice. We use 500 iterations and implement Stochastic Boosting with the *ada* R- package (Culp et al., 2012).

An important parameter in Support Vector Machines is the kernel function. We used the most popular kernels: the linear, polynomial, radial basis (RBF), and sigmoid kernel (Ballings and Van den Poel, 2013a). The RBF and sigmoid kernels require the choice of only one hyper-parameter γ, the width of the Gaussian (Ben-Hur and Weston, 2010). Furthermore, the penalty hyperparameter *C*, also called the cost or soft margin constant, specifies the trade-off between hyperplane violations and the size of the margin. In addition to these parameters the polynomial kernel requires a choice of degree *d*. The linear kernel function only requires setting *C*. One cannot know in advance which settings are best for a given problem. We follow Hsu et al.'s (2010) recommendation to perform a grid search on *C*= $[2^{-5}, 2^{-4}, ..., 2^{15}]$, $\gamma = [2^{-15}, 2^{-13}, ..., 2^{3}]$ and $d = 2, 3$ to identify the best combination. Support Vector Machines are implemented through Meyer et al.'s (2012) *e1071* R-package using the *svm* function.

Ballings and Van den Poel (2013a) recommend the *burn* method for Kernel Factory which automatically selects the best kernel function. Furthermore, we use the recommended values of one column partition and $int(log_{10}(N + 1))$ row partitions. Kernel Factory is implemented using Ballings and Van den Poel's (2013b) *kernelFactory* R-package.

For the feed-forward Artificial Neural Network we use one layer of hidden neurons as it is generally sufficient for classifying most data sets (Dreiseitl and Ohno-Machado, 2002). Before applying the neural network we rescale the numerical predictors to [0,1]. The binary predictors are left untransformed {0,1}. Scaling the data is necessary to obtain training efficiency and overcome numerical problems. The algorithm is implemented using the *nnet* R- package (Ripley, 2013; Venables and Ripley, 2002). The network weights at the start of the iterative procedure are chosen at random (Ripley, 1996, pg. 154). The *entropy* parameter is set to use the maximum conditional likelihood as recommended by Spackman (1991) and Ripley (1996, pg. 149). The *rang* parameter, controlling the range of the initial random weights parameter was left at the default of 0.5. We used weight decay to avoid overfitting (Dreiseitl and Ohno-Machado, 2002). Therefore the maximum number of weights (*MaxNWts*) and the number of iterations (*maxit*) were set to very large values (5000) in order to avoid early stopping. The weight decay factor and the number of nodes in the hidden layer were determined by performing a grid search (Dreiseitl and Ohno-Machado, 2002). We cross-validated all combinations of *decay*={0.001, 0.01, 0.1} (Ripley, 1996, pg. 163), and *size*=[1, 2, ..., 20] (Ripley, 1996, pg. 170) and selected the optimal combination.

4.5.1.2 Combination Algorithms

For comparison purposes we set the number of generations (iterations) to 500 for all general purpose optimization methods. This number is sufficient for all methods to converge. All solutions were repaired by dividing by the sum of all weights.

A Genetic Algorithm has six important parameters: the selection method, population size, mutation chance, cross-over rate, elitism and the number of generations. We implemented the algorithm using the *rbga* function of the *genalg* R package (Willighagen, 2012). The function implements the binary tournament selection method which is efficient and not likely to converge prematurely (Goldberg and Deb, 1991). Harik and Goldberg (1999) developed a population sizing equation for Genetic Algorithms. The disadvantage is that several parameters in the equation cannot be determined directly but need to be estimated (also see Reed et al., 2000). Cormier et al. (2001) use the rule of thumb that a good population size is about seven times the number of variables. As there are six variables this amounts to a population size (*popSize*) of 42. As mentioned above we set the number of required generations to achieve convergence (*maxiter*) to 500. The probability of cross-over is determined by the R-function to be 71%. De Jong (1975) recommends to set the probability of mutation (*pmutation*) to $1/N$, where N is the population size, namely 0.024. We use the default *rbga* setting of 2 for *elitism*, which is 5% of the population size with a minimum of 1. We limited the search space by a maximum individual weight of one and a minimum of zero and constrained the sum of the genes to one. In addition we suggested eight chromosomes. Six of these chromosomes each had a maximum value for one of the six genes. One chromosome had the value of $1/6$ for all genes. We also included a chromosome equal to the authority- based weights. All the other chromosomes were determined at random.

To implement the Differential Evolution Algorithm we use the *DEopt* function of the *NMOF* R- package (Schumann, 2012). Storn and Price (1997) make practical recommendations for the parameters of the Differential Evolutionary Algorithm and Pedersen (2010) refined those recommendations. For our case the optimal parameters for 500 generations are a population size (*nP*) of 20, a probability for cross-over (*CR*) of 0.6938, and a step size (*F*) of 0.9314. Finally, we used the same minimum values, maximum values and suggestions as for the Genetic Algorithm.

Generalized Simulated Annealing is implemented with the *GenSA* R- package (Xiang et al., 2013, 2012). Initial *temperature* is an important control parameter. Kirkpatrick et al. (1983) recommends to set $temperature = \Delta E_{max}$ where ΔE_{max} is the maximum difference of the value of the objective function between solutions (also see Ben-Ameur, 2004). Since we use the AUC as performance criterion this amounts to 0.5. We use the optimal values of Tsallis and Stariolo (1996) for the visiting (*visiting.param*) and acceptance (*acceptance.param*) parameters: respectively 2.7 and -5. We leave the maximum number of calls of the objective function (*max.call*) to the default of 1e7. The maximum number of iterations of the algorithm (*maxit*) is set to 500.

We implemented the Particle Swarm Optimization Algorithm using the *psoptim* function of the *pso* R- package by Bendtsen (2012). We chose the 2011 variant to take into account the latests theoretical advances (Clerc, 2012). The *type* parameter was set to "SPSO2011". We follow all recommended values by Clerc (2012). The maximum number of iterations (*maxit*) is set to 500. The maximum number of function evaluations *maxf* is set to Inf. The absolute convergence tolerance *abstol* is set to -Inf and the tolerance for restarting is zero. The swarm size *s* is 40 and the exponent for calculating the number of informants is set to three. The average percentage of informants for each particle *p* is set to $1 - (1 - 1/s)^k$. Finally, the exploitation constant is $1/(2 * log(2))$, and the local and global exploration (*c.p* and *c.g*) constants are set to $0.5 + log(2)$. The maximum and minimum bounds are identical to the other algorithms.

The Memetic Algorithm with Local Search Chains is implemented by the *malschains* function of the *Rmalschains* R- package (Bergmeir et al., 2013). We follow all recommendations by Molina et al. (2010) for the parameter values. The local search method (*ls*) is set to *cmaes* with the number of iterations of the local search *istep*=300. The *effort* or the ratio between the amount of local and global evaluations is set to 0.5. The cross-over alpha parameter BLX-alpha (*alpha*) is set to 0.5 and the *threshold* which defines how much improvement is considered in the local search as no improvement is set to 10^{-8}. Population size (*popsize*) is 60 individuals. The maximum number of fitness function evaluations *maxEvals* is 500. Finally minimum bounds, maximum bounds and suggestions for the initial population are chosen to be identical to the other algorithms.

We implemented the Self-Organising Migrating Algorithm using the *soma* R-package (Clayden, 2011). We follow the recommendations by Zelinka (2004) for all the parameter values. The distance towards the leader that individuals may migrate to (*pathLength*) is set to 3, the granularity at which potential steps are evaluated (*stepLength*) is set to 0.11, and the probability that individual parameters are changed at any given step (*perturbationChance*) is set to 0.1. We used a value of 0 for the smallest absolute difference between the maximum and minimum cost function values (*minAbsoluteSep*) and a value of 0.001 for the smallest relative difference between the maximum and minimum cost function values (*minRelativeSep*). The maximum number of migrations to complete (*nMigrations*) is set to 500 and the population size (*populationSize*) is evaluated to 10. Finally, the minimum and maximum bounds are identical to the other algorithms.

The Tabu Search Algorithm is implemented using the *tabuSearch* R- package. A crucial parameter is the tabu list size. Values that are too small lead to cycling and values that are too large lead to declaring appealing moves as forbidden (Fouskakis and Draper, 2002). The latter forces the algorithm to explore lower quality solutions and requires a larger number of iterations. In general, larger problems require larger list sizes, but there is no single rule that gives good results for all problems (Fouskakis and Draper, 2002). We follow the recommendations by Glover (1986) to cross-validate the *listSize* from 5 to 12. Analogously to the other algorithms we used 500 iterations (*iters*). Because the package is designed to optimize binary strings we had to convert the solutions to real vectors.

Finally, it is important to note that the three statistical methods Goldfarb-Idnani Non-Negative Least Squares (Goldfarb and Idnani, 1983), Lawson-Hanson Non-Negative Least Squares (Lawson and Hanson, 1987), and Non-Negative Binomial Likelihood (Sra et al., 2008) require no parameters.

4.5.2 Data

To benchmark the Hybrid Ensemble we use eleven data sets from the UCI Machine Learning Repository (Frank and Asuncion, 2010). We selected those data sets because (1) the response variable is binary, (2) there is a good mix of categorical and continuous predictors, (3) most of them are used in key studies such as Breiman's (2001) Random Forest and, (4) they vary in classification difficulty. Table 4.3 gives a summary of the data sets. *N* denotes the number of observations and *n* is the number of predictors.

Table 4.3: Characteristics of the data sets used in the empirical study

Data	N	n, continuous	n, categorical
Ionosphere	351	33	0
Credit	690	6	9
Sonar	208	59	0
Wdbc (Cancer)	569	30	0
HeartHun (Hungary)	294	5	7
GermanCredit	1000	7	13
AustralianCredit	690	6	8
HorseColic	368	9	13
Breast-cancer	286	4	5
Liver	345	6	0
Heart-statlog	270	7	6

4.5.3 Model performance evaluation

To assess the performance of the models we use the area under the receiver operator characteristic curve (AUC or AUROC). AUC is an objective performance metric for classification models (Provost et al., 1998) and is superior to accuracy (percentage correctly classified) in that it is insensitive to the cut-off value applied to the posterior probabilities (Thorleuchter and Van den Poel, 2012, 2013) to classify observations. More specifically, while accuracy measures the performance of the model at one specific cut-off value, AUC assesses performance across all possible cut-off values. AUC is defined as follows:

$$AUC = \int_0^1 \frac{TP}{TP+FN} d\frac{FP}{FP+TN} = \int_0^1 \frac{TP}{P} d\frac{FP}{N} \tag{4.1}$$

with TP: True Positives, FN: False Negatives, FP: False Positives, TN: True Negatives, P: Positives, N: Negatives

AUC is is bounded by 0.5 and 1, where the former value denotes that predictions are not better than random, and the latter indicates perfect prediction. In some cases AUC can be lower than 0.5 on the test sample. This is almost always due to overfitting leading the model to generalize poorly.

All reported AUCs are medians over five replications of twofold cross-validation (5x2 cv) (Dietterich, 1998; Alpaydin, 1999). In each replication all data instances are randomly assigned to one of two parts that are equal in size. Each part is employed as both a training and test set. The entire process results in ten AUCs. The same splits are used for all models. As a measure of dispersion we use the inter quartile range.

In order to test for significant differences we follow the suggestions of Demsar (2006) to use the Friedman test (Friedman, 1937, 1940) along with the Bonferroni-Dunn (Dunn, 1961) post-hoc test. The Friedman test controls the family-wise error, the probability of at least one false positive in any of the comparisons. It is a non-parametric equivalent of the repeated-measures ANOVA and consists in ranking the classifiers to be compared within each data set. We follow the recommendation by Demsar (2006) to start from the median AUCs over 5x2 cv. Next, the best classifier receives rank 1, and the worst classifier receives rank equal to the number of compared classifiers when there are no ties. In case of ties average ranks are assigned (Demsar, 2006). Subsequently, median ranks are computed across the data sets resulting in a final rank per classifier. The Friedman test then checks if these median ranks are significantly different from the average rank expected under the null-hypothesis (Demsar, 2006). Since the primary objective of this paper is to investigate whether the main ensemble is better than the Single Best sub-ensemble (i.e., the control) we use the Bonferroni-Dunn test. Salzberg (1997) notes that this test is very conservative since it assumes the independence of hypotheses. This will allow us to draw stringent conclusions. Classifiers perform significantly different if their median ranks differ by at least the critical difference. The critical difference (CD) is defined as follows (Demsar, 2006):

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{4.2}$$

with q_α the critical value for a given p-value and number of classifiers, *k* the number of classifiers, and *N* the number of data sets. In this study the critical difference for a p-value of 0.05, 13 classifiers, 11 data sets and a critical value of 2.865 equals 4.758.

4.6 Discussion of results

4.6.1 Performance of the Hybrid Ensemble compared to the Single Best

Table 4.4 contains the median AUC of five times twofold cross-validation. The letters in the SB column are the Single Best classifiers per fold. Results in boldface denote the top performing method per data set. It is immediately clear that there is no method that consistently dominates all the others. However, the primary objective of this paper is to find a method that consistently outperforms the Single Best. To this end Figure 4.1 and 4.2 give a graphical overview of the results in Table 4.4. The horizontal axis is the Single Best while the vertical axis is the AUC of the respective competing methods. The points in the plots denote the data sets. If the competing method is consistently (i.e., for all the data sets) better than the Single Best then all points would lie above the dashed diagonal line. Two such methods exist: SOMA and AUTH. The maximum incremental AUC that those methods bring over the Single Best is respectively 0.022 and 0.025, which is sizable given the effective data- strategy the Single Best uses.

The final ranks (lower is better) of the classifiers confirm these findings (see Table 4.5). Both AUTH and SOMA hold the lowest and second lowest ranks (SOMA shares its second position with MEAN, but MEAN does not consistently outperform the SB on all data sets). The Friedman $\chi^2(12)$ is 46.92, with $p < 0.001$. The methods that are significantly better (have ranks lower by an amount equal or larger than the critical difference) than the Single Best are AUTH, SOMA, MEAN, GSA, PSO and NNBL. The stability of the results, measured by the inter quartile range of the AUCs (see Table 4.6), does not differ from method to method. There is no method that consistently has less stable results.

4.6.2 Performance improvement by classification difficulty

One can expect that the higher the performance of the Single Best sub-ensemble, the more difficult it is for the Hybrid Ensemble to increase performance. For example, if a sub-ensemble has an AUC of 0.98 and hence captures most of the diversity through data perturbation, it is very difficult to capture the remaining diversity through algorithm variation. This hypothesis is supported by the data. Admittedly, a sample of 11 data sets does not allow to apply any statistical analyses but there is a clear pattern when the Single Best performance is plotted against the incremental performance by the Hybrid Ensemble. Figure 4.3 clearly shows that higher performances of the Single Best go hand in hand with lower incremental performance by the Hybrid Ensemble (in this case the AUTH method was plotted).

4.6.3 Analysis of the ensemble size

In Section 4.4.1 we have established that the sub-ensembles included in the Hybrid Ensemble are diverse and top performing. However, one might raise questions about the optimality of the

Table 4.4: Median AUC of five times twofold cross-validation

Data	GA	DEA	GSA	MALSC	PSO	SOMA	TSA	LHNNLS	GINNLS	NNBL	MEAN	AUTH	SB	
Ion	0.9743	0.972	0.9765	**0.9779**	0.9748	0.974	0.9772	0.9563	0.976	0.9771	0.9713	0.9765	0.9722	RNRRRRRKKR
Cre	0.9181	0.9191	0.9204	0.9186	0.9193	0.9196	0.9154	0.9132	0.9181	0.9187	0.9207	**0.9222**	0.9162	ARALARRASL
Son	0.8956	0.893	0.8904	0.8929	0.8921	**0.9025**	0.8983	0.8839	0.8997	0.8995	0.9005	0.9025	0.8984	RKKRNRRNRR
Wdb	0.9955	0.9949	0.9957	**0.9958**	0.9948	0.9951	0.9951	0.9899	0.9936	0.9943	0.9952	0.9952	0.9926	LNSLNNLNSL
Hea	0.9042	0.9004	**0.905**	0.903	0.9043	0.8994	0.894	0.883	0.8964	0.9024	0.8924	0.895	0.8936	LARLNLLNLN
Ger	0.7628	0.7642	0.763	0.7629	0.763	0.7637	0.7622	0.7447	**0.7646**	0.7578	0.7611	0.7631	0.7475	LRSLASLRSL
Aus	0.9144	0.9177	0.9136	0.9196	0.9198	0.9211	0.9182	0.9067	0.9235	**0.9266**	0.9217	0.9223	0.9209	NRARRNRLRA
Hor	0.8014	0.7992	0.8024	0.8004	**0.8069**	0.8023	0.7979	0.7805	0.7953	0.8025	0.8039	0.8045	0.7952	RKRRLRLARR
Bre	0.6588	0.6682	0.669	0.6648	**0.6733**	0.6713	0.6619	0.6685	0.6611	0.6679	0.6668	0.6677	0.6538	AKRKRLRLLR
Liv	0.752	0.7483	0.7603	0.7581	0.7612	0.7685	0.7385	0.7482	0.7599	0.7555	**0.7739**	0.7731	0.7619	RNRNLAALAN
Hea	0.892	0.8926	0.896	0.8937	0.8933	0.8939	0.8924	0.8868	0.8918	0.8893	0.894	**0.8969**	0.8721	NLLSRLLSRS

Base Classifiers: S:SVM, R:Random Forest, K:Kernel Factory, L:Logit, A:AdaBoost, N:Neural Network. Best performers per data set are in bold face.
Combination methods: GA: Genetic Algorithm, DEA: Differential Evolutionary Algorithm, GSA: Genaralized Simulated Annealing, MALSC: Memetic Algorithm with Local Search Chains, PSO: Particle Swarm Optimization, SOMA: Self-Organizing Migrating Algorithm, TSA: Tabu Search Algorithm, LHNNLS: Lawson-Hanson Non-Negative Least Squares, GINNLS: Goldfarb-Idnani Non-Negative Least Squares, NNBL: Non-Negative Binomial Likelihood, MEAN: Simple mean, AUTH: Authority- based method, SB: Single Best

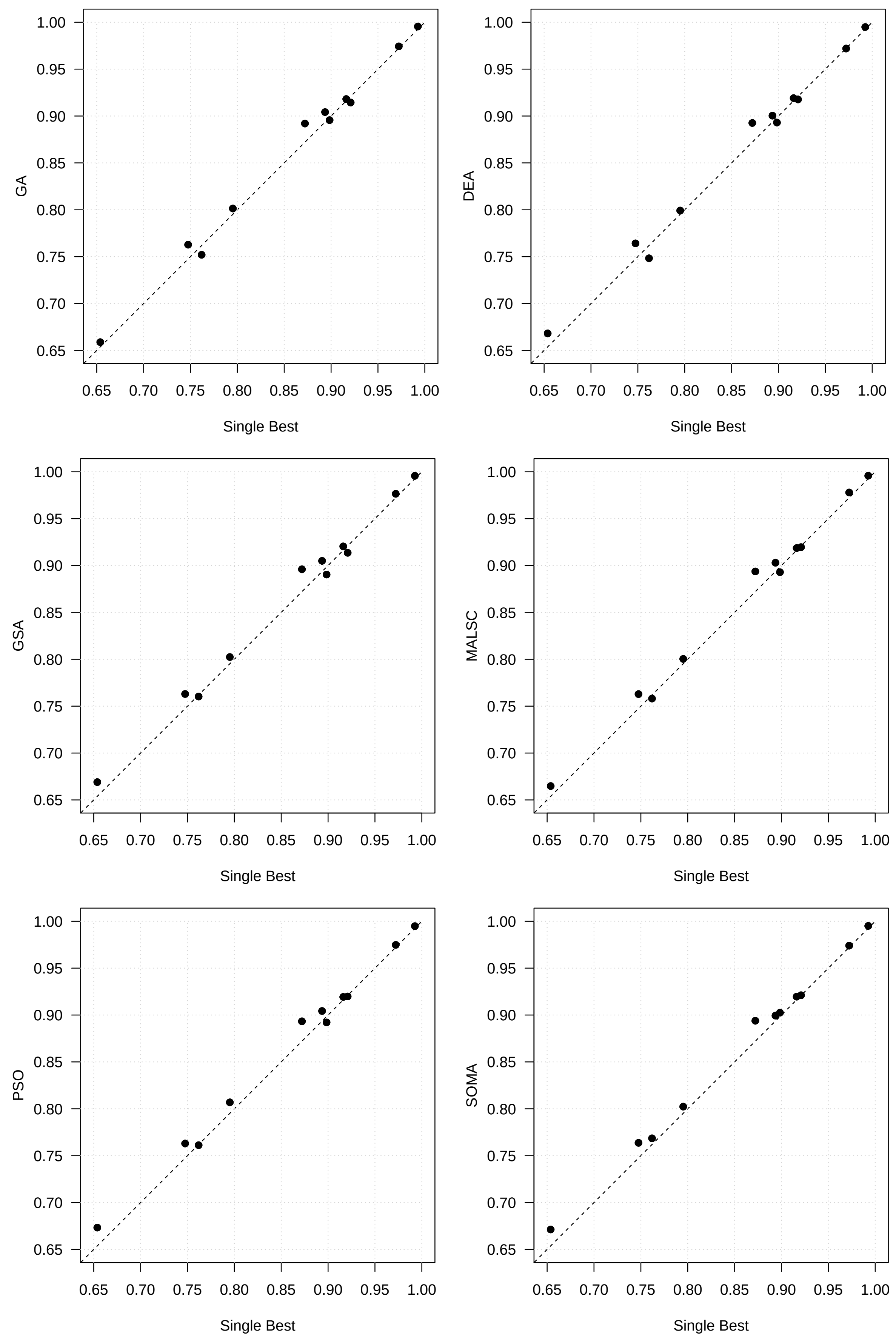


Figure 4.1: Plots comparing the AUC of the Hybrid Ensemble given a specific combiner and the Single Best

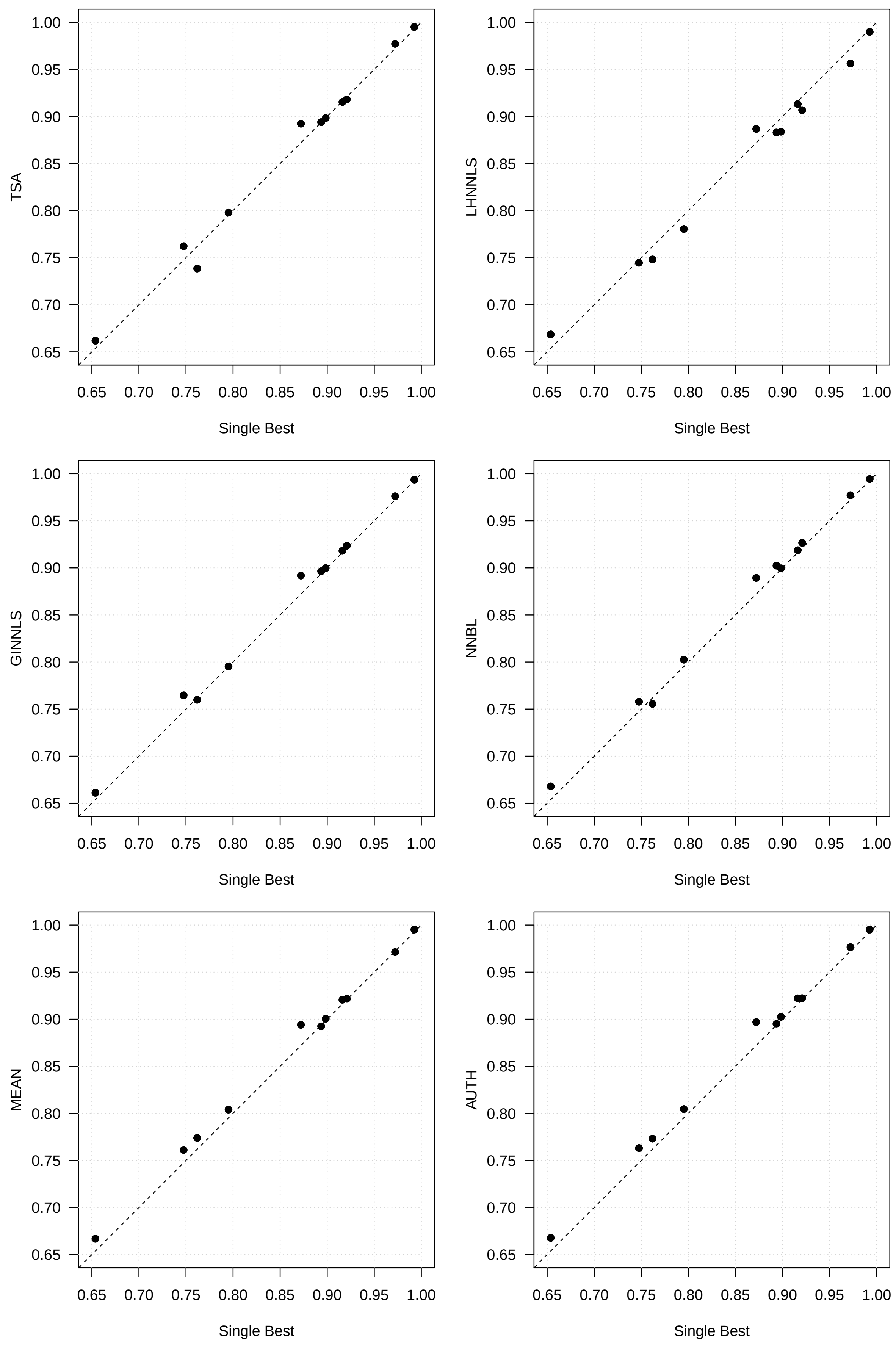


Figure 4.2: Cont'd: Plots comparing the AUC of the Hybrid Ensemble given a specific combiner and the Single Best

Table 4.5: Median ranks across data sets

GA	DEA	GSA	MALSC	PSO	SOMA	TSA	LHNNLS	GINNLS	NNBL	MEAN	AUTH	SB
8.0	8.0	**4.5**	8.0	**5.5**	**4.0**	9.0	13.0	8.0	**6.0**	**4.0**	**3.0**	11.0

Significant differences with SB are in bold face. Combination methods: GA: Genetic Algorithm, DEA: Differential Evolutionary Algorithm, GSA: Genaralized Simulated Annealing, MALSC: Memetic Algorithm with Local Search Chains, PSO: Particle Swarm Optimization, SOMA: Self-Organizing Migrating Algorithm, TSA: Tabu Search Algorithm, LHNNLS: Lawson-Hanson Non-Negative Least Squares, GINNLS: Goldfarb-Idnani Non-Negative Least Squares, NNBL: Non-Negative Binomial Likelihood, MEAN: Simple mean, AUTH: Authority- based method, SB: Single Best

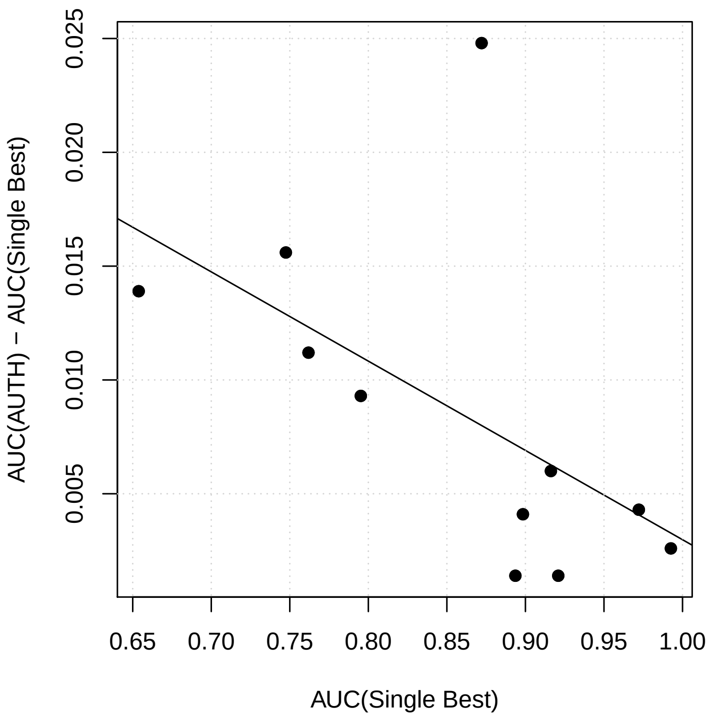


Figure 4.3: Performance improvement through Hybrid Ensemble by difficulty

ensemble size. Why do we include only six sub-ensembles? First of all, six base classifiers already pose a heavy workload for commodity computer hardware, both in the member generation and fusion phase and we do not recommend adding any more at the risk of rendering the analysis intractable (also see Margineantu and Dietterich (1997) for a similar recommendation). Second, near-optimal performance has been observed for small ensembles in extant literature (Margineantu and Dietterich, 1997). Third, diversity plays a key role in ensemble performance when using less than ten classifiers (Kuncheva, 2004; Canuto et al., 2007). Fourth, the purpose of this paper is to improve upon industry's best practice (the Single Best method), hence we included only the top performing ensembles in extant literature.

We have performed a small analysis to confirm the above arguments from extant literature. Figure 4.4 and 4.5 provide an overview of the weights of the top performing combination method that trains weights (SOMA). When a sub-ensemble does not add any value to the total

Table 4.6: IQR AUC of five times twofold cross-validation

Data	GA	DEA	GSA	MALSC	PSO	SOMA	TSA	LHNNLS	GINNLS	NNBL	MEAN	AUTH	SB
Ion	0.0133	0.0154	0.0162	0.0104	0.0093	0.0148	0.0108	0.0583	0.0101	0.0114	0.0121	0.0101	0.0117
Cre	0.0178	0.0118	0.0251	0.0079	0.0244	0.0148	0.0157	0.0581	0.0124	0.0236	0.0212	0.0211	0.0161
Son	0.0396	0.0279	0.0309	0.0284	0.0253	0.0441	0.0319	0.0305	0.0437	0.0307	0.0351	0.0336	0.0252
Wdb	0.006	0.0062	0.0062	0.0053	0.0051	0.0071	0.0061	0.0084	0.0055	0.0066	0.0066	0.0065	0.0077
Hea	0.0204	0.0311	0.0221	0.0235	0.0302	0.0152	0.0256	0.0407	0.0212	0.0317	0.019	0.0207	0.0405
Ger	0.0374	0.0392	0.0316	0.0354	0.0368	0.0343	0.0423	0.0403	0.0346	0.0149	0.0273	0.028	0.0386
Aus	0.025	0.0274	0.0301	0.0235	0.0256	0.0205	0.0227	0.0429	0.0258	0.02	0.02	0.0195	0.0171
Hor	0.0179	0.0205	0.0158	0.0171	0.0203	0.0145	0.0262	0.079	0.015	0.022	0.0308	0.0265	0.0167
Bre	0.0508	0.043	0.0541	0.0438	0.0568	0.0523	0.0577	0.0264	0.0291	0.0443	0.036	0.0424	0.0516
Liv	0.0579	0.0653	0.032	0.0264	0.0333	0.0212	0.0521	0.0343	0.0359	0.0682	0.0283	0.0298	0.0525
Hea	0.0375	0.0299	0.0288	0.0336	0.0317	0.0387	0.0339	0.0548	0.0325	0.0352	0.0268	0.0279	0.0374

Base Classifiers: S:SVM, R:Random Forest, K:Kernel Factory, L:Logit, A:AdaBoost, N:Neural Network

Combination methods: GA: Genetic Algorithm, DEA: Differential Evolutionary Algorithm, GSA: Genaralized Simulated Annealing, MALSC: Memetic Algorithm with Local Search Chains, PSO: Particle Swarm Optimization, SOMA: Self-Organizing Migrating Algorithm, TSA: Tabu Search Algorithm, LHNNLS: Lawson-Hanson Non-Negative Least Squares, GINNLS: Goldfarb-Idnani Non-Negative Least Squares, NNBL: Non-Negative Binomial Likelihood, MEAN: Simple mean, AUTH: Authority based method, SB: Single Best prediction its weight is set to zero. Hence investigating how many of the six weights is zero (or close to zero) gives us an idea about how many sub-ensembles is sufficient. Figure 4.4 plots the weight of each sub-ensemble per data set. In nine out of the eleven data sets at least one weight is lower than 5%. In two data sets even two weights are lower than 5%. Figure 4.5 is an alternative display of the same idea. The weights are sorted from high to low. From the fifth weight on we see weights dropping below the 5% line. The sixth sub-ensemble does only deliver a contribution that is higher than 5% in two data sets. Ultimately, the choice to add more ensemble members is a trade-off between computational efficiency and performance. Given these arguments, we feel that six is a good ensemble size. The question remains open though which other algorithms that are top performing and diverse, could be added to, or replace a member in, the proposed Hybrid Ensemble. Figure 4.4 shows that none of the sub-ensembles consistently performs either well or poorly. This could also be observed from the Single Best column in Table 4.4. Whether an algorithm performs well depends on the data and hence there is no clear indication as to whether an algorithm should be replaced. Nevertheless, that research question is beyond the scope of this paper.

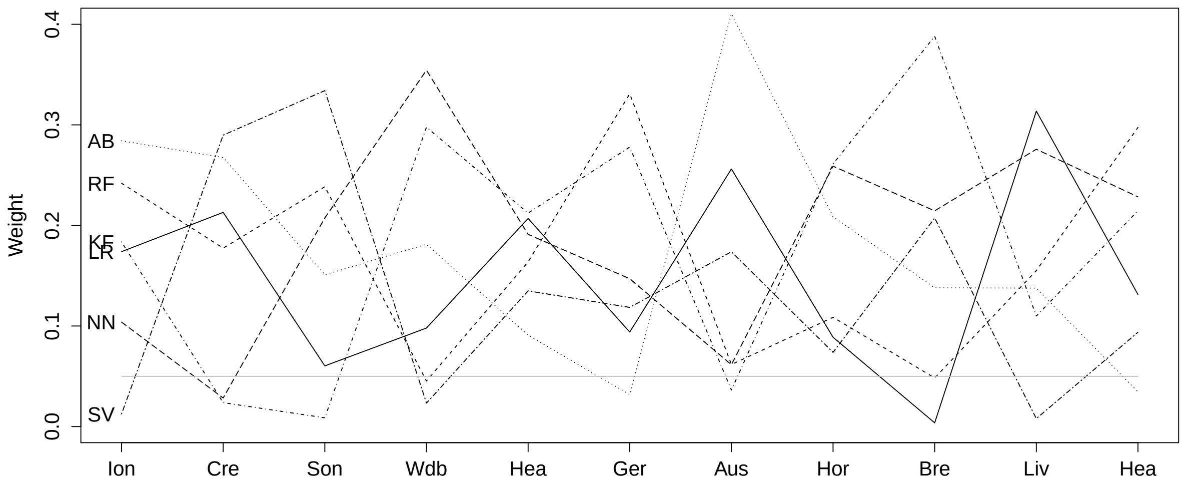


Figure 4.4: Weights per data set

4.7 Conclusions

In this paper we propose a new Hybrid Ensemble consisting of six sub-ensembles. The sub-ensembles generate diversity through data perturbation. Hence the difference of the performance of the Hybrid Ensemble and the Single Best sub-ensemble effectively yields the added value of algorithm-induced diversity. We find that the proposed Hybrid Ensemble gives better performance than the Single Best on all tested data sets for the authority- based weight estimation method and the Self-Organising Migrating Algorithm (SOMA) based weight estimation.

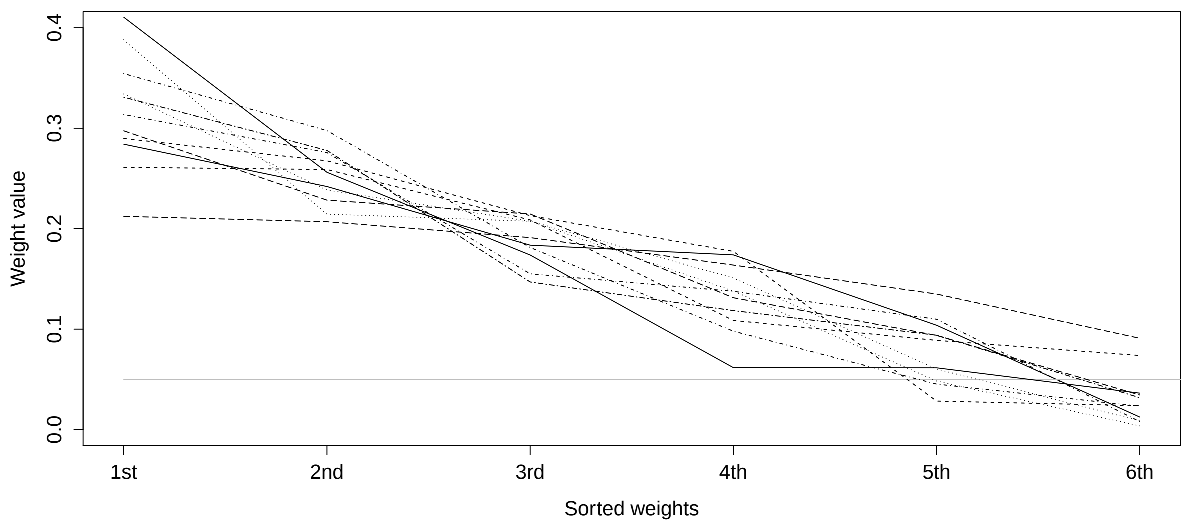


Figure 4.5: Sorted weights

Since the authority- based method outperforms SOMA and in addition is relatively computationally more efficient we recommend it as the default option in future research.

We also found that the added value of the Hybrid Ensemble over the Single Best increases with increasing classification difficulty of the task at hand. An interesting avenue for future research may be to test the proposed method on more difficult tasks and analyze the relationship between difficulty and increased performance in more detail.

Our study gives an indication that six sub-ensembles is sufficient to constitute the Hybrid Ensemble. In addition there is no clear winning or losing sub-ensemble. These findings indicate the appropriateness of the selected algorithms. Future research should provide a decisive answer to these questions, since they are beyond the scope of this paper.

To limit the computational effort, we chose to restrict ourselves to linear combination rules. An interesting path for future research is to evaluate other combination rules including non-linear ones. We submitted the R- software package *hybridEnsemble* to CRAN. The software is open- source and well-documented.

4.8 Acknowledgments

Funding for this research was partially provided by the Special Research Fund (BOF, Bijzonder Onderzoeksfonds), Ghent University, Belgium.

4.9 References

Alpaydin, E., 1999. Combined 5 x 2 cv f test for comparing supervised classification learning algorithms. Neural Computation 11 (8), 1885–1892.

Ballings, M., Van den Poel, D., 2013a. Kernel factory: An ensemble of kernel machines. Expert Systems with Applications 40 (8), 2904–2913.

Ballings, M., Van den Poel, D., 2013b. R package kernelFactory: an ensemble of kernel machines.

Bauer, E., Kohavi, R., 1999. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36 (1-2), 105–139.

Baumgartner, D., Serpen, G., 2012. A design heuristic for hybrid classification ensembles in machine learning. Intelligent Data Analysis 16 (2), 233–246.

Ben-Ameur, W., 2004. Computing the initial temperature of simulated annealing. Computational Optimization and Applications 29, 369–385.

Ben-Hur, A., Weston, J., 2010. A user's guide to support vector machines. Methods in Molecular Biology. Department of Computer Science Colorado State University, pp. 223–239.

Bendtsen, C., 2012. R package pso: Particle swarm optimization.

Bergmeir, C., Molina, D., Benítez, J., 2013. R package rmalschains: Continuous optimization using memetic algorithms with local search chains (MA-LS-Chains) in r.

Berkson, J., Sep. 1944. Application of the logistic function to bio-assay. Journal of the American Statistical Association 39 (227), 357–365.

Breiman, L., Aug. 1996. Bagging predictors. Machine Learning 24 (2), 123–140.

Breiman, L., Oct. 2001. Random forests. Machine Learning 45 (1), 5–32.

Brown, G., Kuncheva, L., 2010. Good and Bad diversity in majority vote ensembles. In: El Gayar, N., Kittler, J., Roli, F. (Eds.), Proceedings 9th International Workshop, Multiple Classifier Systems, MCS 2010. Springer Verlag, Berlin, Germany, pp. 124–33, 7-9 April 2010, Cairo, Egypt.

Brown, G., Wyatt, J., Harris, R., Yao, X., Mar. 2005. Diversity creation methods: a survey and categorisation. Information Fusion 6 (1), 5–20.

Bryll, R., Gutierrez-Osuna, R., Quek, F., Jun. 2003. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognition 36 (6), 1291–1302.

Burez, J., Van den Poel, D., 2009. Handling class imbalance in customer churn prediction. Expert Systems with Applications 36 (3), 4626–4636.

Canuto, A. M. P., Abreu, M. C. C., Oliveira, L. d. M., Xavier, J. C., Santos, A. d. M., Mar. 2007. Investigating the influence of the choice of the ensemble members in accuracy and diversity of selection-based and fusion-based methods for ensembles. Pattern Recognition Letters 28 (4), 472–486.

Caruana, R., Niculescu-Mizil, A., 2006. An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on Machine learning. ICML '06. ACM, New York, NY, USA, p. 161–168.

Chen, D., Plemmons, R. J., 2007. Nonnegativity constraints in numerical analysis. In: Bultheel, A., Cools, R. (Eds.), Proc. Symposium on the Birth of Numerical Analysis. World Scientific Press, Leuven, Belgium.

Cho, S. B., Apr. 1999. Pattern recognition with neural networks combined by genetic algorithm. Fuzzy Sets and Systems 103 (2), 339–347.

Clayden, J., 2011. R package soma: General-purpose optimisation with the self-organising migrating algorithm.

Clerc, M., 2012. Standard particle swarm optimisation, from 2006 to 2011. Technical Report 2012-09-23 version.

Cooper, G. F., Aliferis, C. F., Ambrosino, R., Aronis, J., Buchanan, B. G., Caruana, R., Fine, M. J., Glymour, C., Gordon, G., Hanusa, B. H., Janosky, J. E., Meek, C., Mitchell, T., Richardson, T., Spirtes, P., 1997. An evaluation of machine-learning methods for predicting pneumonia mortality. Artificial Intelligence in Medicine 9 (2), 107–138.

Cormier, G., Boudreau, R., Theriault, S., Dec. 2001. Real-coded genetic algorithm for bragg grating parameter synthesis. Journal of the Optical Society of America B-Optical Physics 18 (12), 1771–1776.

Cortes, C., Vapnik, V., Sep. 1995. Support-vector networks. Machine Learning 20 (3), 273–297.

Coussement, K., Buckinx, W., Nov. 2011. A probability-mapping algorithm for calibrating the posterior probabilities: A direct marketing application. European Journal of Operational Research 214 (3), 732–738.

Coussement, K., Van den Poel, D., 2008. Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. Expert Systems with Applications 34 (1), 313–327.

Culp, M., Johnson, K., Michailidis, G., 2012. R package ada: an r package for stochastic boosting.

De Jong, K., 1975. An analysis of the behavior of a class of genetic adaptive systems. doctoral dissertation, University of Michigan, Ann Arbor.

de Lima, T. P. F., da Silva, A. J., Ludermir, T. B., 2012. Selection and fusion of neural networks via differential evolution. In: Pavon, J., DuqueMendez, N. D., FuentesFernandez, R. (Eds.), Advances in Artificial Intelligence - Iberamia 2012. Vol. 7637. Springer-Verlag Berlin, Berlin, pp. 149–158.

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

Didaci, L., Fumera, G., Roli, F., 2013. Diversity in classifier ensembles: Fertile concept or dead end? Multiple Classifier Systems. 11th International Workshop, MCS 2013. Proceedings, 37–48.

Dietterich, T. G., 1998. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10 (7), 1895–1923.

Dreiseitl, S., Ohno-Machado, L., Dec. 2002. Logistic regression and artificial neural network classification models: a methodology review. Journal of Biomedical Informatics 35 (5-6), 352–359.

Dunn, O. J., Mar. 1961. Multiple comparisons among means. Journal of the American Statistical Association 56 (293), 52–64.

Elkan, C., 2013. Maximum likelihood, logistic regression, and stochastic gradient training. Tech. rep., University of California at Davis, Davis, CA.

Fouskakis, D., Draper, D., Dec. 2002. Stochastic optimization: a review. International Statistical Review 70 (3), 315–349.

Frank, A., Asuncion, A., 2010. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences.
URL `http://archive.ics.uci.edu/ml`

Freund, Y., Schapire, R., 1996. Experiments with a new boosting algorithm. In: Machine Learning. Proceedings of the Thirteenth International Conference (ICML '96). Bari, Italy, pp. 148–156.

Friedman, J., Hastie, T., Tibshirani, R., Feb. 2010. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33 (1), 1–22.

Friedman, J., Hastie, T., Tibshirani, R., 2013. R package glmnet: Lasso and elastic-net regularized generalized linear models.

Friedman, J. H., Oct. 2001. Greedy function approximation: A gradient boosting machine. Annals of Statistics 29 (5), 1189–1232.

Friedman, J. H., Feb. 2002. Stochastic gradient boosting. Computational Statistics & Data Analysis 38 (4), 367–378.

Friedman, M., Dec. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association 32 (200), 675–701.

Friedman, M., Mar. 1940. A comparison of alternative tests of significance for the problem of m rankings. The Annals of Mathematical Statistics 11 (1), 86–92.

Fumera, G., Roli, F., Jun. 2005. A theoretical and experimental analysis of linear combiners for multiple classifier systems. Ieee Transactions on Pattern Analysis and Machine Intelligence 27 (6), 942–956.

Gal-Or, M., May, J. H., Spangler, W. E., 2005. Using decision tree models and diversity measures in the selection of ensemble classification models. In: Oza, N. C., Polikar, R., Kittler, J., Roli, F. (Eds.), Multiple Classifier Systems. Vol. 3541. Springer-Verlag Berlin, Berlin, pp. 186–195.

Geman, S., Bienenstock, E., Doursat, R., Jan. 1992. Neural networks and the bias variance dilemma. Neural Computation 4 (1), 1–58.

Glover, F., 1977. Heuristics for integer programming using surrogate constraints. Decision Sciences 8, 156–166.

Glover, F., 1986. Future paths for integer programming and links to artificial-intelligence. Computers & Operations Research 13 (5), 533–549.

Glover, F., 1989. Tabu search. 1. ORSA Journal on Computing 1 (3), 190–206.

Glover, F., Aug. 1990a. Tabu search - a tutorial. Interfaces 20 (4), 74–94.

Glover, F., 1990b. Tabu search. II. ORSA Journal on Computing 2 (1), 4–32.

Goldberg, D. E., Deb, K., 1991. A comparative analysis of selection schemes used in genetic algorithms. In: Foundations of Genetic Algorithms. Morgan Kaufmann, San Francisco, pp. 69–93.

Goldfarb, D., Idnani, A., 1983. A numerically stable dual method for solving strictly convex quadratic programs. Mathematical Programming 27 (1), 1–33.

Guisan, A., Edwards, T. C., Hastie, T., Nov. 2002. Generalized linear and generalized additive models in studies of species distributions: setting the scene. Ecological Modelling 157 (2-3), 89–100.

Harik, G., Goldberg, D. E., 1999. The gambler's ruin problem, genetic algorithms, and the sizing of populations. Evolutionary Computation 7 (3), 231.

Ho, T. K., Aug. 1998. The random subspace method for constructing decision forests. Ieee Transactions on Pattern Analysis and Machine Intelligence 20 (8), 832–844.

Ho, T. K., May 2002. Multiple classifier combination: Lessons and next steps. In: Hybrid Methods In Pattern Recognition. Vol. 47. World Scientific, pp. 171–198.

Holland, J. H., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., 2010. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan.

Kim, E., Kim, W., Lee, Y., Jan. 2003. Combination of multiple classifiers for the customer's purchase behavior prediction. Decision Support Systems 34 (2), 167–175.

Kim, Y. S., Jan. 2006. Toward a successful CRM: variable selection, sampling, and ensemble. Decision Support Systems 41 (2), 542–553.

King, R., Feng, C., Sutherland, A., 1995. Statlog - comparison of classification algorithms on large real-world problems. Applied Artificial Intelligence 9 (3), 289–333.

Kirkpatrick, S., Gelatt, C., Vecchi, M., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.

Kittler, J., Hatef, M., Duin, R. P. W., Matas, J., Mar. 1998. On combining classifiers. Ieee Transactions on Pattern Analysis and Machine Intelligence 20 (3), 226–239.

Ko, A. H. R., Sabourin, R., Britto, A. S., May 2008. From dynamic classifier selection to dynamic ensemble selection. Pattern Recognition 41 (5), 1718–1731.

Krogh, A., Vedelsby, J., 1995. Neural network ensembles, cross validation, and active learning. Advances in Neural Information Processing Systems 7, 231–238.

Kuncheva, L., 2004. Combining Pattern Classifiers. Methods and Algorithms. Wiley.

Kuncheva, L. I., 2003. That elusive diversity in classifier ensembles. In: Perales, F. J. (Ed.), Pattern Recognition and Image Analysis, Proceedings. Vol. 2652. Springer-Verlag Berlin, Berlin, pp. 1126–1138.

Kuncheva, L. I., Whitaker, C. J., May 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning 51 (2), 181–207.

Lawson, C. L., Hanson, R. J., 1987. Solving Least Squares Problems. Prentice-Hall, Englewood Cliffs, NJ.

Liaw, A., Wiener, M., 2002. Classification and regression by randomForest. R News 2 (3), 18–22.

Liaw, A., Wiener, M., 2012. R package randomForest: breiman and cutler's random forests for classification and regression.

Lim, T. S., Loh, W. Y., Shih, Y. S., 2000. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning 40 (3), 203–228.

Margineantu, D. D., Dietterich, T. G., 1997. Pruning adaptive boosting. In: Proc. 14th Int'l Conf. Machine Learning. pp. 211–218.

McCulloch, W., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133.

Menahem, E., Shabtai, A., Rokach, L., Elovici, Y., Feb. 2009. Improving malware detection by applying multi-inducer ensemble. Computational Statistics & Data Analysis 53 (4), 1483–1494.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., 2012. R package e1071: Misc functions of the department of statistics (e1071),.

Michalski, R. S., Michalski, R. S., Tecuci, G., Carbonell, J. G., Mitchell, T. M., 1994. Machine Learning: A Multistrategy Approach. Morgan Kaufmann.

Mitchell, T. M., 1997. Machine Learning, 1st Edition. McGraw-Hill, Inc., New York, NY, USA.

Molina, D., Lozano, M., Garcia-Martinez, C., Herrera, F., 2010. Memetic algorithms for continuous optimisation based on local search chains. Evolutionary Computation 18 (1), 27–63.

Murphy, A. H., Winkler, R. L., 1977. Reliability of subjective probability forecasts of precipitation and temperature. Journal of the Royal Statistical Society 26 (1), 41–47.

Myung, I. J., Feb. 2003. Tutorial on maximum likelihood estimation. Journal of Mathematical Psychology 47 (1), 90–100.

Nascimento, D. S. C., Coelho, A. L. V., 2009. Ensembling heterogeneous learning models with boosting. In: Leung, C. S., Lee, M., Chan, J. H. (Eds.), Neural Information Processing, Pt 1, Proceedings. Vol. 5863. Springer-Verlag Berlin, Berlin, pp. 512–519.

Neslin, S. A., Gupta, S., Kamakura, W., Junxiang Lu, Mason, C. H., May 2006. Defection detection: Measuring and understanding the predictive accuracy of customer churn models. Journal of Marketing Research (JMR) 43 (2), 204–211.

Opitz, D. W., Shavlik, J. W., 1996. Generating accurate and diverse members of a neural-network ensemble. In: Touretzky, D. S., Mozer, M. C., Hasselmo, M. E. (Eds.), Advances in Neural Information Processing Systems 8: Proceedings of the 1995 Conference. Vol. 8. M I T Press, Cambridge, pp. 535–541.

Partridge, D., Yates, W. B., May 1996. Engineering multiversion neural-net systems. Neural Computation 8 (4), 869–893.

Pedersen, M., 2010. Good parameters for differential evolution. Technical Report HL1002, Hvass Laboratories.

Perlich, C., Provost, F., Simonoff, J. S., Feb. 2004. Tree induction vs. logistic regression: A learning-curve analysis. Journal of Machine Learning Research 4 (2), 211–255.

Provost, F., Fawcett, T., Kohavi, R., 1998. The case against accuracy estimation for comparing induction algorithms. In: Shavlik, J. (Ed.), Machine Learning. Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98). Morgan Kaufmann Publishers, Madison, WI, USA, pp. 445–453.

R Core Team, R., 2013. R package stats: R statistical functions.

Reed, P., Minsker, B., Goldberg, D. E., 2000. Designing a competent simple genetic algorithm for search and optimization. Water Resources Research 36 (12), 3757–3761.

Ripley, B., 1996. Pattern Recognition and Neural Networks. Cambridge University Press.

Ripley, B., 2013. R package nnet: Feed-forward neural networks and multinomial log-linear models.

Rokach, L., Feb. 2010. Ensemble-based classifiers. Artificial Intelligence Review 33 (1-2), 1–39.

Roli, F., Fumera, G., Kittler, J., 2002. Fixed and trained combiners for fusion of imbalanced pattern classifiers. Int Soc Information Fusion, Sunnyvale.

Ruta, D., Gabrys, B., Mar. 2005. Classifier selection for majority voting. Information Fusion 6 (1), 63–81.

Salzberg, S. L., Nov. 1997. On comparing classifiers: Pitfalls to avoid and a recommended approach. Data Mining and Knowledge Discovery 1 (3), 317–328.

Schumann, E., 2012. R package NMOF: numerical methods and optimization in finance.

Sharkey, A. J. (Ed.), 1999. Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, 1st Edition. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Spackman, K. A., 1991. Maximum likelihood training of connectionist models: comparison with least squares back-propagation and logistic regression. In: Proceedings of the Annual Symposium on Computer Application in Medical Care. pp. 285–289.

Sra, S., Kim, D., Scholkopf, B., 2008. Non-monotonic poisson likelihood maximization. Technical Report 170, Max Planck Institute for Biological Cybernetics.

Storn, R., Price, K., Dec. 1997. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11 (4), 341–359.

Tang, E. K., Suganthan, P. N., Yao, X., Oct. 2006. An analysis of diversity measures. Machine Learning 65 (1), 247–271.

Thorleuchter, D., Van den Poel, D., Dec. 2012. Predicting e-commerce company success by mining the text of its publicly-accessible website. Expert Systems with Applications 39 (17), 13026–13034.

Thorleuchter, D., Van den Poel, D., Apr. 2013. Technology classification with latent semantic indexing. Expert Systems with Applications 40 (5), 1786–1795.

Tsallis, C., Stariolo, D. A., Nov. 1996. Generalized simulated annealing. Physica A 233 (1-2), 395–406.

Tsoumakas, G., Angelis, L., Vlahavas, I., 2005. Selective fusion of heterogeneous classifiers. Intelligent Data Analysis 9 (6), 511–525.

Ueda, N., Feb. 2000. Optimal linear combination of neural networks for improving classification performance. Ieee Transactions on Pattern Analysis and Machine Intelligence 22 (2), 207–215.

Valentini, G., Muselli, M., Ruffino, F., 2003. Bagged ensembles of support vector machines for gene expression data analysis. In: Proceedings of the International Joint Conference on Neural Networks 2003, Vols 1-4. Ieee, New York, pp. 1844–1849.

van der Laan, M. J., Polley, E. C., Hubbard, A. E., Oct. 2007. Super learner. Statistical Applications in Genetics and Molecular Biology 6.

Venables, W. N., Ripley, B. D., 2002. Modern Applied Statistics with S, 4th Edition. Springer, New York.

Wang, W. J., Jones, P., Partridge, D., 2000. Diversity between neural networks and decision trees for building multiple classifier systems. In: Kittler, J., Roli, F. (Eds.), Multiple Classifier Systems. Vol. 1857. Springer-Verlag Berlin, Berlin, pp. 240–249.

Willighagen, E., 2012. R package genalg: R based genetic algorithm.

Windeatt, T., Mar. 2005. Diversity measures for multiple classifier system analysis and design. Information Fusion 6 (1), 21–36.

Woods, K., Kegelmeyer, W. P., Bowyer, K., Apr. 1997. Combination of multiple classifiers using local accuracy estimates. Ieee Transactions on Pattern Analysis and Machine Intelligence 19 (4), 405–410.

Wu, J. X., Zhou, Z. H., Chen, Z. Q., 2001. Ensemble of GA based selective neural network ensembles. Fudan Univ Press, Shanghai.

Xiang, Y., Gubian, S., Suomela, B., Hoeng, J., 2012. Generalized simulated annealing for efficient global optimization: the GenSA package for r. The R Journal Forthcoming.

Xiang, Y., Gubian, S., Suomela, B., Hoeng, J., 2013. R package GenSA: r functions for generalized simulated annealing.

Xu, L., Krzyzak, A., Suen, C., Jun. 1992. Methods of combining multiple classifiers and their applications to handwriting recognition. Ieee Transactions on Systems Man and Cybernetics 22 (3), 418–435.

Yates, W. B., Partridge, D., 1996. Use of methodological diversity to improve neural network generalisation. Neural Computing & Applications 4 (2), 114–128.

Yu, Y., Zhou, Z.-H., Ting, K. M., 2007. Cocktail ensemble for regression. In: Ramakrishnan, N., Zaiane, O. R., Shi, Y., Clifton, C. W., Wu, X. D. (Eds.), Icdm 2007: Proceedings of the Seventh Ieee International Conference on Data Mining. Ieee Computer Soc, Los Alamitos, pp. 721–726.

Zadrozny, B., Elkan, C., 2002. Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining. ACM Press, Edmonton, pp. 694–699.

Zelinka, I., 2004. SOMA — self-organizing migrating algorithm. In: New Optimization Techniques in Engineering. Vol. 141 of Studies in Fuzziness and Soft Computing. Springer, pp. 167–217.

Zhang, B., 2011. Two-stage hybrid classifier ensembles for subcellular phenotype images classification. Procedia Environmental Sciences 8, 554–562.

Zhang, L., Zhou, W.-D., Jan. 2011. Sparse ensembles using weighted combination methods based on linear programming. Pattern Recognition 44 (1), 97–106.

Zhou, Z.-H., 2012. Ensemble Methods: Foundations and Algorithms, 1st Edition. Chapman & Hall/CRC.

Zhou, Z. H., Wu, J. X., Tang, W., May 2002. Ensembling neural networks: Many could be better than all. Artificial Intelligence 137 (1-2), 239–263.