UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Marcelo Antônio Mendes Bastos

Poda de Ensemble via uma abordagem de programação inteira com restrições de diversidade

Belo Horizonte
2021

Marcelo Antônio Mendes Bastos

Poda de Ensemble via uma abordagem de programação inteira com restrições de diversidade

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Cristiano Arbex Valle
Coorientador: Humberto César Brandão de Oliveira

Belo Horizonte
2021

Marcelo Antônio Mendes Bastos

Ensemble pruning via an integer programming approach with diversity constraints

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Cristiano Arbex Valle
Co-Advisor: Humberto César Brandão de Oliveira

Belo Horizonte
2021

Bastos, Marcelo Antônio Mendes.

B327e Ensemble pruning via an integer programming approach with diversity constraints [manuscrito] / Marcelo Antônio Mendes Bastos – 2021.
xx,58 f. il.

Orientador: Cristiano Arbex Valle.
Coorientador: Humberto César Brandão de Oliveira.
Dissertação (mestrado) – Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f.49-53.

1. Computação – Teses. 2. Aprendizagem por Ensemble – Aprendizado do computador – Teses. 3. Programação Inteira – Teses. I. Valle, Cristiano Arbex. II. Oliveira, Humberto César Brandão de. III Universidade Federal de Minas Gerais; Instituto de Ciências Exatas, Departamento de Ciência da Computação. IV. Título.

CDU 519.6*82.10(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Ensemble pruning via an integer programming approach with diversity constraints

MARCELO ANTÔNIO MENDES BASTOS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

[illegible]

Prof. Cristiano Arbex Valle - Orientador
Departamento de Ciência da Computação - UFMG

Humberto Brandão

Prof. Humberto César Brandão de Oliveira - Coorientador
Departamento de Ciências Exatas - UNIFAL

[illegible]

Profa. Gisele Lobo Pappa
Departamento de Ciência da Computação - UFMG

[illegible]

Prof. Túlio Ângelo Machado Toffolo
Departamento de Computação - UFOP

Belo Horizonte, 29 de Março de 2021.

Acknowledgments

First, I would like to thank God, for having always blessed me and given me strength in all moments of this journey.

To my dear parents Antônio and Zelma, and my sister Natália, for all the love and care given, for always supporting and encouraging me in times of difficulty.

To my cousins Rafael and Guilherme, the latter who unfortunately is no longer with us, for inspiring me to get a graduate degree and go to a master's program. Despite of the difficulties, their intelligence and hardworking inspire me to keep studying and give the best of myself.

To my co-advisor professor Humberto for helping to open the doors to I come to this master's program, for introducing me to the areas of machine learning and optimisation and allowing me to work with his initial optimisation model, which gave rise to my undergraduate thesis and following to this master's thesis. More over, his experience really contributed to improve my professional and academic qualifications.

I would like to give a special thanks to my advisor Professor Cristiano, for guiding me through this work, for all the shared knowledge and all the support he gave me. He was a very dedicated advisor, whose contributions really helped to improve the quality this thesis. I really appreciate all the knowledge that I acquire from him during the period of the master's program, it has been of great value in both my academic and professional life.

Finally, I also thank all the friends and other people who helped me in some way, both during the development of this work and in the stages before it.

Resumo

Aprendizagem por ensemble utiliza múltiplos classificadores na expectativa de obter um melhor desempenho preditivo. Estudos empíricos vem mostrando que a poda de ensembles, isto é, a escolha de um subconjunto apropriado dos classificadores disponíveis, pode levar a previsões comparáveis ou melhores do que usar todos os classificadores. Nesta dissertação, consideramos um problema de classificação binária e propomos uma abordagem de programação inteira (PI) para selecionar subconjuntos de classificadores ótimos. Propomos uma função objetivo flexível para se adaptar aos critérios desejados de diferentes conjuntos de dados. Também propomos restrições para garantir níveis mínimos de diversidade no conjunto. Apesar do caso geral de PI ser NP-Difícil, os solvers de última geração são capazes de obter rapidamente boas soluções para conjuntos de dados com até 60.000 pontos de dados. Nossa abordagem produz resultados competitivos quando comparados a outros algoritmos de poda na literatura.

Palavras-chave: Programação Inteira, Poda de Ensemble, Seleção em Ensemble, Classificação

Abstract

Ensemble learning uses multiple classifiers in the hope of obtaining better predictive performance. Empirical studies have shown that ensemble pruning, that is, choosing an appropriate subset of the available classifiers, can lead to comparable or better predictions than using all classifiers. In this thesis, we consider a binary classification problem and propose an integer programming (IP) approach for selecting optimal classifier subsets. We propose a flexible objective function to adapt to desired criteria of different datasets. We also propose constraints to ensure minimum diversity levels in the ensemble. Despite the general case of IP being NP-Hard, state-of-the-art solvers are able to quickly obtain good solutions for datasets with up to 60000 data points. Our approach yields competitive results when compared to others pruning algorithms in literature.

Keywords: Integer Programming, Ensemble Pruning, Ensemble Selection, Classification

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Ensemble learning is a popular technique in the domain of machine learning. An ensemble is defined as the aggregation of multiple classifications into a single final decision. It is generally accepted in literature that the precision of an ensemble tends to improve when compared to the behaviour of individual classifiers [47].

Well known approaches for efficiently generating ensembles include Bagging (bootstrap aggregating) [6] and Boosting [21], the latter having given rise to popular variations such as Random Forests [7] and extreme gradient boosting [11]. A general feature of these approaches is that all classifiers are considered in the aggregation. There are, however, theoretical and empirical studies which have shown that pruning an ensemble by selecting a subset of the classifiers can lead to comparable or better predictions [29; 47]. Two criteria are important when defining the most appropriate subset: the overall performance of the ensemble and how diverse the selected classifiers are amongst themselves.

In this thesis we tackle the ensemble pruning problem by introducing an integer programming (IP) approach for choosing an optimal subset of binary classifiers. Our formulation optimises a weighted function of the patterns in the binary confusion matrix. This provides enough flexibility to ensure suitable optimisation criteria dependent on the properties of the underlying dataset. As our objective is performance-based we also introduce linear constraints that ensure minimum diversity levels in the ensemble. The formulation is original in considering the optimisation of the confusion matrix. To the best of our knowledge our approach has not been proposed in literature before.

Despite several techniques for ensemble pruning having been previously proposed, we believe that our approach contributes to the current knowledge in the field. IP modelling is a flexible tool, adaptable to particularities of different problems. One of the most important advantages in applying this tool to ensemble pruning is being able to combine performance and diversity criteria. Moreover an IP framework provides an exact method, as opposed to most algorithms in literature which are generally suboptimal.

The general IP problem is NP-Hard, however decades of research in algorithmic techniques have led to state-of-the-art solvers which are able to effectively solve several industrial-sized problems. In this thesis we show that with such solvers we can find good solutions to relatively large problems in reasonable computational times.

A problem that may come out while using optimisation to find optimal solutions in machine learning problems is the risk of overfitting: This term is used in machine learning for when the model learns "too much" from a training dataset but loses generality when applied to new data points. For this reason, we use in-sample and out-of-sample sets to validate our formulation regarding possible overfitting. The machine learning algorithms and the formulations will only have access to in-sample data, and its results are validated in out-of-sample data. We use a stratified K-fold cross-validation procedure in order to generate out-of-sample results.

We compare our formulation to a full ensemble and two other well-known methods in literature: Reduced-Error Pruning With Backfitting [22] (performance-based) and Kappa pruning [30] (diversity-based). We report competitive results for publicly available datasets ranging from 195 to 60000 data points.

1.1 Organisation

This thesis is organised as follows. In Chapter 2 we give a brief overview of known methods for the ensemble learning process and describe its three stages: classifier generation, classifier selection and classifier combination. In Chapter 3 we present our optimisation approach. We also amend the formulation with extra constraints to enforce minimum diversity levels.

Our computational experiments are shown in Chapter 4. We present the datasets used in our experiments and we examine both in-sample and out-of-sample performance, comparing it to three different benchmarks. We also test the use of majority voting to define the classification threshold in our formulations.

In Chapter 5 we discuss two cutting-plane reformulations of the problem. This is a topic of ongoing research and we only introduce the formulations, separation algorithms and some implementation details, but we do not include any computational results. The main goal is to attempt to improve computational performance of our optimisation approach. Finally, in Chapter 6 we present our concluding remarks.

Chapter 2

Literature review

The process of ensemble learning is generally composed of three stages: generation, selection and combination of classifiers. In the next sections we describe and give examples for each one of them.

2.1 Classifier generation

The ensemble process starts by generating a set of distinct classifiers. The usefulness of any ensemble depends on classifiers being both precise and diverse. High correlations among classifications may hinder the benefits of combining multiple classifiers.

Several techniques for ensuring diversity in classifiers have been proposed [17; 14]. Here we list some of them:

1. Randomising classifiers: Some classifiers have built in random components. Using different random initialisations in those components may create diverse classifiers.
2. Parameter Tuning: Classification models have their own parameters, and diversity in classification can be increased by initialising models with different settings. Some examples include using different pruning levels in decision trees or exploring different architectures in multi-layer perceptron neural networks.
3. Combining distinct classifier models: The combination of distinct classifiers models (e.g. Decision Trees, Neural Networks and SVM) into heterogeneous ensembles is also a popular approach for ensuring diversity.
4. Different distributions of the training set: Diversity is achieved by using different samples of the training data to construct each classifier. Some examples are the aforementioned Bagging and Boosting techniques.

5. Distinct features subsets: Diversity is accomplished by using distinct subsets of features (e.g. Random Subspace Method [24; 41]) or different feature extraction methods (e.g. extraction of facial image features [4]).

In this thesis, we do not focus on classifier generation. Our approach is general given any set of previously selected classifiers.

2.2 Classifier selection

The second stage in the ensemble learning process is the selection of a subset of classifiers to generate the final ensemble, a process known as ensemble pruning. This is optional since it is possible to use the full set of generated classifiers to compose the ensemble. However, as previously mentioned, there are theoretical and empirical studies which have shown that pruning an ensemble can be advantageous [29; 47].

The selection process can be static or dynamic. In dynamic selection, different subsets are chosen for different data points. In static selection, only one subset of classifiers is chosen for all data points.

The reasoning behind dynamic selection is that certain subsets of classifiers are more specialised in different parts of the feature space [49]. Generally clustering algorithms are used for selecting multiple subsets. In this case, initially the data points are divided into different clusters and then subsets of classifiers are selected for each cluster. An unseen data point is first assigned to one of the clusters and then classified by the respective subset of classifiers previously selected for that cluster. For more details we refer the reader to [8; 14].

Our approach is based on static selection. Static selection policies can be ranking based, cluster based and optimisation based [44]. In the following subsections we will detail those policies. [44] refer to static selection as ensemble pruning, the same terminology we adopt throughout this text.

2.2.1 Ranking based methods

Ranking based methods sort classifiers according to a fitness function. In general they are computationally fast but suboptimal.

In Kappa pruning [30], every pair of classifiers is sorted according to a statistical measure of agreement, named k-statistic. The M pairs with the lowest agreement levels are selected, where M is the parameter that specify the ensemble size.

Reordering techniques [31] in bagging classifiers have been used to build subensembles of increasing size, adding first those classifiers that are expected to perform best when combined.

A dynamic programming approach was proposed by [15] to improve computational efficiency of ranking based methods, where it was integrated in a classical ranking-based algorithm. Experimental results demonstrate significantly efficiency improvements acquired with the integration.

2.2.2 Cluster based methods

Cluster based methods first apply a clustering algorithm to group similar classifiers and then prune each cluster separately to increase general diversity.

There are some issues to consider in these methods. A first one is which clustering algorithm to use. Known clustering algorithms include k-means [27; 40], hierarchical agglomerative clustering [23] and deterministic annealing [2].

An important issue is the measure of distance between clusters. Different distance measures have been used such as Euclidean distance in training set [27; 40]. [23] proposed a measure based on an estimation of the probability that two classifiers make the same mistakes in a separated validation set.

Another issue to consider is the number of clusters, a necessary parameter in some clustering algorithms, like k-means. [40] determined it by analysing the performance on a validation set and [27] gradually increased this number until the disagreement between the cluster centroids stopped improving.

The remaining issue is the cluster pruning policy. [23] selected from each cluster the classifier which is most distant from the others. [27] used an approach where the models are iteratively removed in increasing order of accuracy, until the accuracy of the ensemble starts to decrease. [40] selected the most accurate model of each cluster. In [2], a new model is trained for each cluster, using the cluster centroids as values of the target output.

2.2.3 Optimisation based methods

Optimisation based methods attempt to find a subset that optimises a given performance metric. Most methods found in literature offer approximate solutions due to optimising non-linear metrics.

Hill climbing has been the most popular heuristic approach for ensemble pruning. It is a local search algorithm that starts with an arbitrary solution to a problem and makes incremental changes in the solution in order to find better ones. In the ensemble pruning problem, hill climbing starts with an arbitrary subset of classifiers and constructs a new subset by adding or removing classifiers from it. The metrics used in hill climbing ensemble pruning approaches can be grouped in two categories: performance based and diversity based.

Regarding performance based measures, accuracy was used in [30; 18; 9]. Additionally to accuracy, a metric called Benefit was also used in [18]. In [9], several metrics were tested: Accuracy, mean cross-entropy, lift, precision/recall break-even point, precision/recall F-score, average precision, ROC Area, a measure of probability calibration and SAR.

In diversity based methods, [42] tested several diversity metrics: disagreement, double fault, Kohavi-Wolpert variance, inter-rater agreement, generalized diversity and difficulty. [33; 3; 37; 39] introduced diversity metrics specially designed for hill climbing.

A genetic algorithm was developed by [48]. The ensemble is represented as a string of bits, where the number 1 means that the classifier is present in the ensemble, and 0 otherwise. The authors used the common operations of genetic algorithms, as mutation and crossover. The Fitness function is the error rate of the selected subensemble of classifiers.

Mathematical programming approaches have also been proposed. [47] formulated the problem as a quadratic integer programming problem, aiming to optimise a trade-off between diversity and accuracy. They used semi-definite programming to find approximate solutions to their formulation. [46] also employed a mathematical formulation to optimise a trade-off between diversity and accuracy. Starting with an integer programming formulation, they relax and reformulate it as a constrained eigenvector problem, which can be solved with an efficient algorithm that is guaranteed to converge globally.

2.2.4 Others

Other approaches include using statistical methods for heterogeneous ensembles, reinforcement learning and a boosting based procedure.

[43; 45] use statistical tests to prune heterogeneous ensembles. The tests are used to measure the significance of the prediction performances between the ensemble classifiers. The classifiers with higher and statistically significant performance remain in the ensemble.

Reinforcement learning is used by [36; 38]. The problem of selecting a subset of an ensemble with n classifiers is modelled as an episodic task with n time steps in each episode. An episode starts with an empty set of classifiers, and in each time step it takes the action of include or exclude the corresponding classifier in the final ensemble. The episodes iteration stops when the algorithm used to approximate the optimal policy of their reinforcement learning problem converges.

[32] use a boosting based procedure. The authors use an iterative algorithm that at each iteration selects the classifier with the lowest weighted error on the training set. The initialisation and updating of the instance weights happens similar to the AdaBoost algorithm. The difference is that the process is not finished when the weighted error is larger than 0.5 as in AdaBoost, but the instance weights are reset and the model selection continues. It continues until the desired number of selected classifiers (given by a parameter) is achieved.

2.3 Classifier combination

The last stage is the process of combining the classifiers output. The outputs are combined by a combination rule, generating the output of the final ensemble.

In discrete classification problems the most common approach for combining classifiers is through majority voting, where the output of a given data point is assigned to the most voted class among the ensemble classifiers. In probability based classifications, it is generally computed averages in the output of the classifiers.

Weights may be assigned to individual classifiers according to their importance in the ensemble composition. For instance, in the weighted voting, one or more classifiers can be counted multiple times during the voting, receiving more importance in the final decision. The weighted average can also be used, where more important classifiers receive higher weights during the combination. For further details we refer the reader to [26].

Chapter 3

An integer programming model for ensemble pruning

In this chapter we present our optimisation model, detailing our integer programming formulation in Section 3.1 and restrictions created to enforce minimum diversity levels in Section 3.2.

3.1 Formulation

Consider a binary classification problem where data points belong to classes 1 (positive) or 0 (negative). In this section we present an IP formulation for choosing an optimal subset of binary classifiers.

Let $\mathcal{K} = \{1, \ldots, K\}$ be the set of classifiers. Let $\mathcal{N}_0 = \{1, \ldots, N_0\}$ and $\mathcal{N}_1 = \{1, \ldots, N_1\}$ be the sets of negative and positive data points respectively, with $N = N_0 + N_1$ being the total number of data points. Consider a $N_1 \times K$ matrix B where $\beta_{ik} = 1$ if classifier $k \in \mathcal{K}$ correctly classified data point $i \in \mathcal{N}_1$ as positive, $\beta_{ik} = 0$ if it mistakenly classified i as negative. Accordingly consider a $N_0 \times K$ matrix A where $\alpha_{jk} = 0$ if classifier $k \in \mathcal{K}$ correctly classified data point $j \in \mathcal{N}_0$ as negative, $\alpha_{jk} = 1$ if j was mistakenly classified as positive.

Suppose $\mathcal{S} \subseteq \mathcal{K}$ is a set of S classifiers selected to compose a given pruned ensemble. For any data point $i \in \mathcal{N}_1$, $\sum_{s \in \mathcal{S}} \beta_{is}$ is the number of correct positive classifications within $\mathcal{S}$. Accordingly, for any data point $j \in \mathcal{N}_0$, $\sum_{s \in \mathcal{S}} \alpha_{js}$ represents the number of (wrong) positive classifications within $\mathcal{S}$.

We define a threshold $0 \leq L \leq S$ such that for a given data point $i \in \mathcal{N}_1$, $\sum_{s \in \mathcal{S}} \beta_{is} > L$ implies that the ensemble classifies i as positive. If $\sum_{s \in \mathcal{S}} \beta_{is} \leq L$, then i is classified by the ensemble as negative. Similarly for $j \in \mathcal{N}_0$, $\sum_{s \in \mathcal{S}} \alpha_{js} > L$ implies a positive ensemble classification and $\sum_{s \in \mathcal{S}} \alpha_{js} \leq L$ implies a negative ensemble classification. For instance, if $S = 10$ and $L = 5$, then the ensemble classifies a data point as positive if at least 6 individual classifications are positive. If 5 or less are positive, then the ensemble classifies that data point as negative.

In our formulation we let the optimisation define both $\mathcal{S}$ and L. Hence we include L as a general integer variable representing the classification threshold and binary variables $x_k = 1$ if classifier $k \in \mathcal{K}$ is chosen to compose the ensemble ($x_k = 0$ otherwise).

		Predicted Class	
		1	0
Actual Class	1	T^+	F^-
	0	F^+	T^-

Table 3.1: Binary classification confusion matrix

Consider the binary confusion matrix given in Table 3.1. For each of the four possible patterns we assign weights $W_T^+, W_T^-, W_F^+, W_F^- \in \mathbb{R}$. The objective function is defined by the weighted sum $W_T^+T^+ + W_F^-F^- + W_T^-T^- + W_F^+F^+$.

For modelling this weighted sum we define further binary variables t_i^+, f_i^- if the ensemble classification of $i \in \mathcal{N}_1$ is respectively a true positive or false negative. Similarly we define binary variables t_j^-, f_j^+ if the ensemble classification of $j \in \mathcal{N}_0$ is a true negative or false positive. The IP formulation that optimises a weighted sum of the patterns in the binary confusion matrix is given below:

$$\max \sum_{i=1}^{N_1} (W_T^+ t_i^+ + W_F^- f_i^-) + \sum_{j=1}^{N_0} (W_T^- t_j^- + W_F^+ f_j^+) \tag{3.1}$$

subject to

$$(L+1) - \sum_{k=1}^{K} x_k\, \beta_{ik} \leq (K+1)(1 - t_i^+), \qquad \forall i \in \mathcal{N}_1 \tag{3.2}$$

$$\sum_{k=1}^{K} x_k\, \beta_{ik} - L \leq (K+1) t_i^+, \qquad \forall i \in \mathcal{N}_1 \tag{3.3}$$

$$t_i^+ + f_i^- = 1, \qquad \forall i \in \mathcal{N}_1 \tag{3.4}$$

$$\sum_{k=1}^{K} x_k\, \alpha_{jk} - L \leq K(1 - t_j^-), \qquad \forall j \in \mathcal{N}_0 \tag{3.5}$$

$$(L+1) - \sum_{k=1}^{K} x_k\, \alpha_{jk} \leq K t_j^-, \qquad \forall j \in \mathcal{N}_0 \tag{3.6}$$

$$f_j^+ + t_j^- = 1, \qquad \forall j \in \mathcal{N}_0 \tag{3.7}$$

$$x_k \in \mathbb{B} \qquad \forall k \in \mathcal{K} \tag{3.8}$$

$$t_i^+, f_i^- \in \mathbb{B} \qquad \forall i \in \mathcal{N}_1 \tag{3.9}$$

$$t_j^-, f_j^+ \in \mathbb{B} \qquad \forall j \in \mathcal{N}_0 \tag{3.10}$$

$$0 \leq L \leq K, \tag{3.11}$$

$$L \in \mathbb{Z} \tag{3.12}$$

Constraints (3.2) ensure that a positive data point $i \in \mathcal{N}_1$ has $t_i^+ = 1$ if the number of individual positive classifications exceeds L. Conversely, Constraints (3.3) ensure that $t_i^+ = 0$ if the number of individual positive classifications is no more than L. Constraints (3.4) ensure that either $t_i^+ = 1$ or $f_i^- = 1$. Constraints (3.5) guarantee that a negative data point $j \in \mathcal{N}_0$ has $t_j^- = 0$ if the number of positive classifications exceeds L. Otherwise, constraints (3.6) make sure that $t_j^- = 1$. Constraints (3.7) ensure that either $f_j^+ = 1$ or $t_j^- = 1$. Constraints (3.8-3.12) define variables bounds.

3.1.1 Objective function

For some classification problems, optimising some patterns may be prioritised instead of others. Take, for instance, an investment decision problem where investing in the wrong project may cause bankruptcy while not investing into a promising project may prove to be a lost opportunity (regretful, but not as serious). In this case we need to prioritise the minimisation of F^+ at the expense of others. Observe that in order to minimise a pattern of the confusion matrix, we need to set its respective weight as negative. In fact, as F^+ and F^- represents the misclassifications, we will usually minimise them or set their weights to zero.

The weights in Equation (3.1) provide flexibility for defining optimisation criteria depending on the characteristics of the dataset at hand (such as being highly imbalanced). A few examples of possible criteria are outlined below.

Accuracy is defined as $\frac{T^+ + T^-}{N}$. As N is constant we can maximise accuracy by defining weights $W_T^+ = W_T^- = 1$ and $W_F^+ = W_F^- = 0$. Notice that if we choose this objective then constraints (3.3) and (3.6) are redundant as maximising positive weights W_T^+ and W_T^- ensure that $t_i^+ = 1$ and $t_i^- = 1$ if allowed by constraints (3.2) and (3.5). Similarly recall is defined as $\frac{T^+}{T^+ + F^-} = \frac{T^+}{N_1}$ and can be maximised by setting $W_T^+ = 1$ and $W_T^- = W_F^+ = W_F^- = 0$ (with Constraints (3.3) being redundant).

Now consider $\theta = \frac{N_1}{N}$ as an imbalance parameter of the dataset. If, for instance, $\theta \leq 1 - \epsilon$ for small ϵ, then a high accuracy can be obtained by simply classifying every data point as positive. For such cases a possibly useful configuration of the objective function is obtained by setting weights $W_T^+ = (1 - \theta)$, $W_T^- = \theta$ and $W_F^+ = W_F^- = 0$. This may provide a good approximation for more appropriate criteria for imbalanced datasets such as precision, balanced accuracy and the area under the ROC (Receiver Operating Characteristics) curve, best known as AUC [19].

3.2 Diversity

As mentioned before many ensemble pruning algorithms are based on diversity measures. Our proposed formulation optimises a performance measure, and in this section we introduce a way to control diversity with linear constraints.

We consider a diversity measure called Pairwise Failure Crediting (PFC), proposed originally by **(author?)** [10] and chosen due to past experience having achieved satisfactory performance in imbalanced datasets [5; 20]. PFC measures how diverse an individual classifier is from the remaining classifiers in the ensemble.

PFC is calculated as follows. For each classifier k, we compute a failure pattern (FP). A FP is a string of 0's and 1's with length N. A '0' in the string means that the classifier failed to correctly predict the corresponding data point and a '1' means that it predicted the data point correctly (irrespective of its real value).

Once we have all failure patterns we take any two classifiers k and l and calculate their Hamming distance. The Hamming distance between same-length strings is the number of different characters in the same positions. For example, if $\mathrm{FP}_k = \{0011011101\}$ and $\mathrm{FP}_l = \{0110001110\}$, the Hamming distance between k and l is 5 (characters 2, 4, 6, 9 and 10 differ).

Next, we sum all failures by both classifiers - that is, we sum the number of zeros in both strings which, in the example, is 9. The failure credit (FC) between k and l is obtained by dividing the Hamming distance by the sum of failures. In the example, $\mathrm{FC}_{kl} = 5/9$. For every pair $k, l \in \mathcal{K}$ we compute FC_{kl}.

Consider again $\mathcal{S}$ as a set of $S \leq K$ classifiers that are selected to compose an ensemble. We assume without loss of generality that classifiers in $\mathcal{S}$ are indexed by $k = 1, \ldots, S$. Their PFC values are given by:

$$\mathrm{PFC}_k = \frac{\sum_{l=1, l\neq k}^{S} \mathrm{FC}_{kl}}{S-1} \qquad k \in S$$

A (maximum) value of 1 in PFC_k means that k classifies all data points differently from every other classifier in the ensemble, and a (minimum) value of 0 means that k is identical to all other classifiers. Both extreme cases imply that all other classifiers are identical among themselves.

For ensuring minimum desired diversity levels, we propose two approaches:

- The minimum PFC value of any individual classifier is at least a certain threshold $0 \leq \tau \leq 1$ in order to prevent very similar pairs of classifiers.
- The average PFC value of the ensemble must be at least a certain threshold $0 \leq \gamma \leq 1$ to ensure an overall good level of diversity. Clearly we must have $\gamma \geq \tau$.

We add the following new decision variables. Let $y_{kl} = 1$ if both classifiers k and l have been selected to be part of the ensemble, and $y_{kl} = 0$ if at most one of k and l is chosen to compose the ensemble. This adds $\binom{K}{2}$ extra variables (for every possible pair k, l). For simplicity, both y_{kl} and y_{lk} denote the exact same variable. The following constraints ensure that y_{kl} takes the correct values:

$$y_{kl} \geq x_k + x_l - 1 \qquad \forall k, l \in \mathcal{K}, k < l \tag{3.13}$$

$$y_{kl} \leq x_k \qquad \forall k, l \in \mathcal{K}, k < l \tag{3.14}$$

$$y_{kl} \leq x_l \qquad \forall k, l \in \mathcal{K}, k < l \tag{3.15}$$

$$y_{kl} \geq 0 \qquad \forall k, l \in \mathcal{K}, k < l \tag{3.16}$$

Notice that there is no need for the y_{kl} variables to be binary. As both x_k and x_l are binary, y_{kl} must have integer values in any integer solution.

We can then rewrite the PFC equation using variables x_k and y_{kl}:

$$\mathrm{PFC}_k = \frac{\sum_{l=1, l \neq k}^{K} \mathrm{FC}_{kl}\, y_{kl}}{\sum_{m=1}^{K} x_m - 1} \qquad \forall k \in \mathcal{K}$$

The term $\sum_{m=1}^{K} x_m$ is the cardinality of the ensemble and any non-selected classifier k (with $x_k = 0$) has a PFC value of 0 (as all $y_{kl} = 0, l \neq k$).

The following linear constraints ensure that every classifier has a minimum PFC value of τ:

$$\sum_{\substack{l=1 \\ l \neq k}}^{K} \mathrm{FC}_{kl}\, y_{kl} \geq \tau \Big(\sum_{m=1}^{K} x_m - 1 \Big) - K\tau(1 - x_k) \qquad \forall k \in \mathcal{K} \tag{3.17}$$

The term $K\tau(1 - x_k)$ ensures that the constraints above are only enforced if classifier k is chosen to compose the ensemble.

The following nonlinear constraint ensures that the average PFC of the ensemble is at least γ:

$$\frac{1}{\sum_{m=1}^{K} x_m} \; \frac{\sum_{k=1}^{K} \sum_{l=1, l \neq k}^{K} \mathrm{FC}_{kl}\, y_{kl}}{\sum_{m=1}^{K} x_m - 1} \geq \gamma \tag{3.18}$$

Observe that in Equation (3.18) the FC value of every pair is added twice. We use this fact to linearise this expression. For a given subset $\mathcal{S}$, the average PFC μ_{PFC} is given by:

$$\begin{aligned}
\mu_{\text{PFC}} &= \frac{1}{S}\sum_{k=1}^{S}\frac{\sum_{l=1,l\neq k}^{S}\text{FC}_{kl}}{S-1} \\
&= \frac{1}{S(S-1)}\sum_{k=1}^{S}\sum_{\substack{l=1\\ l\neq k}}^{S}\text{FC}_{kl} \\
&= \frac{2}{S(S-1)}\sum_{k=1}^{S-1}\sum_{l=k+1}^{S}\text{FC}_{kl} \\
&= \frac{1}{\binom{S}{2}}\sum_{k=1}^{S-1}\sum_{l=k+1}^{S}\text{FC}_{kl} = \mu_{\text{FC}}
\end{aligned}$$

where μ_{FC} denotes the average FC value of all pairs in the ensemble. We conclude that $\mu_{FC} = \mu_{PFC}$, that is, the average PFC among all classifiers in the ensemble is equal to the average FC among all pairs.

If S classifiers are selected in the ensemble, then the number of y_{kl} variables that take value 1 is exactly $\binom{S}{2}$. Therefore we can ensure that the average PFC value is at least γ with the following linear constraint:

$$\sum_{k=1}^{K-1}\sum_{l=k+1}^{K}\text{FC}_{kl}\, y_{kl} \geq \gamma \sum_{k=1}^{K-1}\sum_{l=k+1}^{K} y_{kl} \tag{3.19}$$

The expanded formulation with minimum diversity levels is given by maximising (3.1) subject to (3.2)-(3.17) and (3.19). It requires $\binom{K}{2}$ extra variables and a similar number of extra constraints, which could lead it to be more computationally demanding. However we show in Section 4.2 that the inclusion of such constraints causes a negligible decrease in solution quality.

Chapter 4

Computational experiments

In this chapter we outline the computational experiments used to evaluate the proposed formulation. In Section 4.1, we present the datasets used in our experiments. In Section 4.2 we examine the performance and solution quality of the formulation in a few selected test sets. As previously mentioned the general case of IP formulations is NP-Hard, but advances in solution techniques have yield effective solvers that are able to quickly find solutions of high quality for several IP problems. In this section we intend to verify whether that is true for our proposed formulation by analysing in-sample solutions.

Finally, in Section 4.3 we conduct out-of-sample experiments using a stratified 10-fold cross-validation procedure, comparing our formulation with 3 benchmarks, using the Accuracy (Subsection 4.3.2), AUC (Subsection 4.3.3) and Balanced Accuracy (Subsection 4.3.4) as evaluation metrics.

For the experiments in this Chapter, we prepared 10 different heterogeneous classifier models. Each model was instantiated a number of times with different random seeds and parameters. We set K as multiples of 10 in order to have an equal number of instantiations of each classifier. For instance, if $K = 60$, we have 6 classifiers of each model. In our experiments, we used $K = \{10, 20, 30\}$ for the experiments in Section 4.2 and $K = \{40, 60, 80, 100\}$ for experiments in Section 4.3. Each classifier produces, as output, a probability of a data point being positive. This probability is rounded to define matrices A and B. A more thorough description of the classifiers can be found in Appendix A.

4.1 Datasets

For computational experimentation we chose 9 publicly available datasets. All datasets can be found at the UCI Machine Learning Repository [28]. They range from $N = 195$ to $N = 60000$ datapoints. We also include columns with the number of features, the number of datapoints in Class 0 (N_0) and Class 1 (N_1), the value of the imbalance parameter θ ($\theta = N_1/N$) as well as the meaning of each class in the context of each dataset. Table 4.1 lists all datasets.

Table 4.1: Selected datasets from the UCI Machine Learning Repository [28]

Identifier	Dataset name	Features	N	N_0	N_1	θ	Class 0	Class 1
PRK	Parkinsons	23	195	48	147	0.77	Healthy	Parkinson's disease
MSK	Musk (Version 1)	168	476	269	207	0.44	Non-musk	Musk
BCW	Breast Cancer Wisconsin	32	569	357	212	0.37	Benign cancer	Malignant cancer
QSR	QSAR biodegradation	41	1055	356	699	0.66	Ready biodegradable	Not ready biodegradable
DRD	Diabetic Retinopathy Debrecen	20	1151	540	611	0.53	No signs of diabetic retinopathy	Signs of diabetic retinopathy
SPA	Spambase	57	4601	2788	1813	0.39	Not spam	Spam
DEF	Default of credit card clients	24	30000	23364	6636	0.22	Default payment (no)	Default payment (yes)
BMK	Bank Marketing	21	41188	36548	4640	0.11	Not subscribed to a term deposit	Subscribed to a term deposit
APS	APS Failure at Scania Trucks	171	60000	59000	1000	0.02	Not related to the APS system failure	APS system specific component failure

Dataset PRK aim is to discriminate healthy people from those with Parkinsons disease. Dataset MSK describes molecules judged by human experts to be musks and new molecules are predicted to be musks or non-musks. In dataset BCW the goal is to predict whether data points represent benign or malignant cancer from features extracted from digitized images. In dataset QSR, the task is to predict whether a molecule is ready or not ready biodegradable. The goal of dataset DRD is to decide whether an image contains signs of diabetic retinopathy or not. Dataset SPA attempts to identify spam e-mails. Dataset DEF examines Taiwanese credit information to decide whether customers are credible (low chance of default) or not. Dataset BMK consists of predicting whether a client subscribed to a certain bank product or not. Dataset APS aims to predict if a Failure in the Air Pressure system (APS) at Scania Trucks is due to a failure in a specific component from the APS or due to components not related to the APS.

4.2 Solving the formulation

0-1 IP is notoriously computationally difficult. Its feasibility version is one of Karp's 21 NP-complete problems [25]. Modern day solvers however can effectively solve large instances of several different IP problems by employing a combination of techniques (such as branch-and-bound and cutting planes) for search space reduction. In this section we verify whether that is true for the formulation proposed when solving the datasets outlined in Section 4.1. Our goal is to check whether optimal or good enough solutions can be found within reasonable computational times.

For all the experiment in this section and this thesis in general, we employed CPLEX 12.8 [13] with default parameters as the IP solver and we run all experiments in an Intel Core(TM) I7-7700 @ 3.60GHz with 32GB of RAM, using 8 cores and having Linux as the operating system.

In all experiments in this section we set weights $W_T^+ = W_T^- = 1$ and $W_F^+ = W_F^- = 0$, which as discussed earlier is equivalent to maximising accuracy. We test here two versions of our formulation: **F1**, which employs only constraints (3.2)-(3.12) and **F3** which also employs minimum diversity constraints. In **F3** we set $\tau = \text{PFC}_{\text{min}}$ and $\gamma = \frac{\text{PFC}_{\text{min}}+\text{PFC}_{\text{avg}}}{2}$, where PFC_{min} and PFC_{avg} are the minimum individual PFC and the average PFC value of the full ensemble. So F1 does not make use of diversity constraints while F3 does. More details on these configurations are given below in Section 4.3.2, where we also define another configuration **F2**.

We run two different sets of experiments. In the first, we set a maximum time limit of 1 hour. Here our intention is to evaluate whether optimal solutions are found within this limit or, if not, how far we are from proving optimality. We set the total number of classifiers as $K = 10, 20, 30$.

In the second set of experiments, we set a time limit of only 5 minutes and increase the number of classifiers to $K = 100$. Since most methods reviewed in literature are suboptimal requiring usually short computational times, we want to evaluate if commercial solvers can "compete" with them when considering larger instances.

Table 4.2 presents the results considering a time limit of 1 hour. In this first set of experiments, each classifier was initially trained with 70% of its original data points, while the remaining 30% points were classified and used to generate the A and B matrices, which were then used by our formulation to choose the subset of classifiers that maximised accuracy. In the table, column $\mathbf{N'}$ shows the number of data points (30% of the original dataset) and column $\mathbf{K}$ shows the number of classifiers.

For F1 we include four columns. T(s) denotes the total time elapsed in seconds, LB and UB respectively represent the best solution found (lower bound) and best upper bound obtained at the end of the search, either when the instance was solved to proven optimality or when the time limit was reached. GAP is defined as $100(\text{UB} - \text{LB})/\text{UB}$. An empty value under T(s) means that the time limit of 1 hour was reached before proving optimality. Empty values for both UB and GAP mean that the solution was solved to optimality within the time limit.

For F3, we also include, under the Diff column, the relative difference in terms of solution quality between F1 and F3. Here we intend to test the effect of adding diversity constraints in the quality of in-sample solutions. Since F3 is more constrained, its optimal solution can only be equal or worse than F1.

Dataset	N'	K	**F1**				**F3**					Best classifier	Diff	Full ensemble	Diff
			T(s)	LB	UB	GAP	T(s)	LB	UB	GAP	Diff				
PRK	59	10	0.0	55	-	-	0.0	55	-	-	0.00	52	-5.45	50	-9.09
		20	0.0	56	-	-	0.0	56	-	-	0.00	53	-5.36	51	-8.93
		30	0.0	56	-	-	0.1	56	-	-	0.00	53	-5.36	50	-10.71
BCW	171	10	0.0	165	-	-	0.0	165	-	-	0.00	165	0.00	163	-1.21
		20	0.0	165	-	-	0.0	165	-	-	0.00	165	0.00	165	0.00
		30	0.0	166	-	-	0.0	166	-	-	0.00	165	-0.60	164	-1.20
MSK	143	10	0.0	131	-	-	0.0	131	-	-	0.00	131	0.00	128	-2.29
		20	0.0	134	-	-	0.0	134	-	-	0.00	132	-1.49	128	-4.48
		30	0.0	134	-	-	0.1	134	-	-	0.00	132	-1.49	128	-4.48
QSR	317	10	0.0	281	-	-	0.0	281	-	-	0.00	276	-1.78	274	-2.49
		20	0.1	285	-	-	0.1	285	-	-	0.00	281	-1.40	273	-4.21
		30	0.1	285	-	-	0.1	285	-	-	0.00	281	-1.40	274	-3.86
SPA	1381	10	0.1	1333	-	-	0.1	1333	-	-	0.00	1328	-0.38	1320	-0.98
		20	0.8	1335	-	-	2.1	1335	-	-	0.00	1328	-0.52	1322	-0.97
		30	1.6	1336	-	-	10.8	1336	-	-	0.00	1328	-0.60	1317	-1.42
DRD	346	10	0.4	252	-	-	0.6	252	-	-	0.00	251	-0.40	238	-5.56
		20	18.0	264	-	-	9.1	264	-	-	0.00	257	-2.65	240	-9.09
		30	118.4	265	-	-	86.2	265	-	-	0.00	257	-3.02	238	-10.19
DEF	9000	10	30.2	7392	-	-	24.0	7392	-	-	0.00	7384	-0.11	7376	-0.22
		20	-	7401	7479.2	1.06	-	7397	7494.2	1.31	-0.05	7384	-0.23	7369	-0.43
		30	-	7404	7534.0	1.76	-	7399	7541.6	1.93	-0.07	7384	-0.27	7370	-0.46
BMK	12357	10	60.0	11364	-	-	76.1	11363	-	-	-0.01	11363	-0.01	11321	-0.38
		20	-	11377	11620.2	2.14	-	11377	11647.2	2.38	0.00	11363	-0.12	11321	-0.49
		30	-	11382	11720.0	2.97	-	11378	11711.1	2.93	-0.04	11363	-0.17	11307	-0.66
APS	18000	10	0.3	17904	-	-	1.0	17904	-	-	0.00	17899	-0.03	17888	-0.09
		20	3.9	17917	-	-	5.7	17917	-	-	0.00	17906	-0.06	17891	-0.15
		30	12.2	17920	-	-	29.8	17920	-	-	0.00	17906	-0.08	17892	-0.16

Table 4.2: In-sample results

In the last four columns, we calculate the objective function value (number of correct guesses) for the best classifier among the K chosen and for an ensemble composed of all classifiers. The latter was calculated by setting $x_k = 1$ to every classifier k. In both these cases we set $L = K/2$. In other words, majority voting was used to choose the appropriate ensemble prediction.

We also include the corresponding Diff column for each. Notice that the optimal solution of F1 must always be equal or better than both these benchmarks, since both benchmarks are feasible solutions to the F1 itself.

Within 1 hour, the solver used was able to optimally solve 23 out of 27 different instances, for both F1 and F3. In the worst case, for BMK with $K = 30$, the gap between the best solution and best upper bound was was 2.97% after one hour.

All solutions also had higher accuracy than the benchmarks in all cases. As previously mentioned this was expected for the cases where optimality was proven (as the formulation maximises accuracy), however it also happened for the larger instances which were not proven to optimality. We observed that for the experiments we run, the solver with its default configurations generally found quickly solutions of high quality.

An interesting aspect we observed, not reported in the table, was that the linear relaxation values after solving the first node were stronger for the larger instances than for the smaller instances. We suspect that as more data points imply more constraints in the formulation (albeit also adding extra dimensions), the convex hull is better approximated. We leave a more thorough analysis of these observations for further work.

We also note that the addition of diversity constraints in **F3** caused a negligible decrease in performance, even considering the addition of $O(\binom{K}{2})$ extra variables and constraints. In some cases the time required to find the optimal solution was even faster. Also, the optimal solutions were either equal or very slightly worse than those of F1.

In the second set of in-sample experiments, we consider a time limit of 5 minutes, with $K = 100$. Here we configure the experiments differently. We used 27% of the total data points instead of 30% and we run each experiment 100 times, each execution varying both the set of 27% data points selected as well as random seeds of each classifier. We used the same configuration that was employed in the out-of-sample experiments, for which a detailed explanation is given in Section 4.3. As mentioned before the goal of these results is to verify how close to optimality solvers can achieve within a short time limit.

Table 4.3 summarises the average optimality gaps among 100 executions, as well as their respective standard deviations. The largest instance, APS, had average gaps of only 0.1% in both configurations F1 and F3. The hardest instance in F1 was DEF (6.7% and 7.0% average gaps). The inclusion of diversity constraints made a noteworthy difference for the DRD instance (4.4% and 7.1% average gaps), making it the hardest instance in F3.

Table 4.3: Average optimality gaps and corresponding standard deviations (in %).

Instance	27% of N	**F1**		**F3**	
		Avg.	Std.	Avg.	Std.
PRK	53	0.0	0.0	0.0	0.0
BCW	154	0.0	0.0	0.0	0.0
MSK	129	0.0	0.0	0.0	0.0
QSR	285	0.0	0.0	0.0	0.1
DRD	311	4.4	2.1	7.1	2.0
SPA	1242	0.0	0.0	0.2	0.2
DEF	8100	6.7	0.4	7.0	0.4
BMK	11121	5.4	0.3	5.6	0.3
APS	16200	0.1	0.0	0.1	0.0

In our view, even the hardest instances were still relatively close to optimality considering the low computational effort. For this reason, in all computational experiments reported hereafter we set a time limit of 5 minutes. If the time limit is reached, we halt the solver and retrieve the best available solution at that point.

4.3 Out of sample

In the previous section we established that, for the instances tested, optimal or good enough solutions can generally be found within a short time. In practice by setting a short time limit we are essentially treating an exact algorithm as a heuristic, such as most methods in literature which are suboptimal in nature (Section 2.2). In this section we evaluate whether the advantages of an exact approach (possibility of optimal solutions, flexibility of an IP model) are helpful regarding out-of-sample performance.

For evaluating performance we used a stratified 10-fold cross-validation procedure. The N data points are initially shuffled randomly and the dataset is then split into 10 folds. At each iteration, one of the folds is left out as an independent set (unseen by the algorithms). The results presented below are based solely in this set. The other 9 folds, comprising 90% of the original dataset, are joined and further split into two sets: a training set, containing 63% of the data points, is used to train the individual classifiers. A validation set, comprising the remaining 27% data points, is used to optimise the ensemble selection algorithms. The procedure is illustrated in Figure 4.1.

The procedure above is repeated 10 times: in each we vary the random seeds required to both shuffle the dataset and initialise the individual classifiers. For each value of K and for each instance shown in Table 4.1, we run 100 experiments: 10 random initialisations $\times$ 10 folds.

4.3.1 Benchmarks

In order to evaluate our formulation, we compare it with three other approaches: Full ensemble (without any pruning), Reduced-Error Pruning With Backfitting [22] (hereby referred to as Backfitting) and Kappa pruning [30]. All three benchmarks classify data points based on majority voting.

Backfitting follows a greedy approach with revision. Its objective is maximising accuracy. Initially, the classifier subset S is empty. Then, at each iteration, the algorithm adds to S a classifier s such that the accuracy of $S \cup s$ is maximised. This process repeats until a predefined number M of classifiers is added to S. Ties are broken arbitrarily. Whenever a classifier is added, the greedy choice is revised through a local search procedure. Each classifier in the ensemble is iteratively replaced by another previously left out. If the overall accuracy is improved, the method starts again from the new subset S. The local search stops after 100 iterations or whenever no classifier is able to improve the

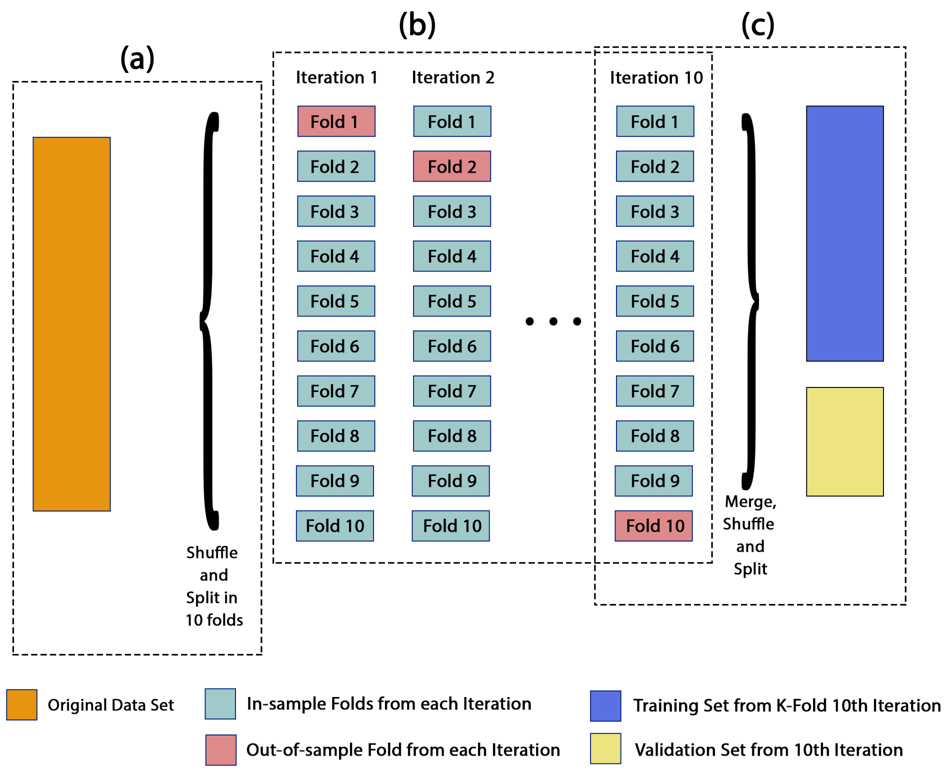
(a)
(b)
(c)
Shuffle and Split in 10 folds
Iteration 1
Iteration 2
Iteration 10
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Fold 6
Fold 7
Fold 8
Fold 9
Fold 10
Merge, Shuffle and Split
Original Data Set
In-sample Folds from each Iteration
Out-of-sample Fold from each Iteration
Training Set from K-Fold 10th Iteration
Validation Set from 10th Iteration

Figure 4.1: K-Fold procedure. (a) Original dataset is shuffled and split in 10 folds. (b) At each iteration, 1 fold is left out as the out-of-sample independent set, and the other 9 as in-sample folds. (c) The in-sample folds are merged, shuffled and split in two sets: the Training set used in the Machine learning stage, and validation set used in the optimisation stage. In the figure this procedure is illustrated for the 10th iteration.

solution, whichever happens first.

Kappa pruning follows a similar procedure, but with two notable differences: there is no revision of the greedy choice and, instead of maximising accuracy, it optimises a diversity measure. The fitness function is the κ-statistic [12], which is a measure of statistical agreement between any two classifiers.

As opposed to our formulation, both Backfitting and Kappa pruning require the ensemble subset size to be fixed. To allow a fairer comparison, we varied M within 20% and 80% of K. The best in-sample results are used to evaluate the independent set. In line with our choice of time limit for the IP formulation, both algorithms are allowed to run for a maximum of 5 minutes.

4.3.2 Accuracy

In our first suite of experiments we set weights $W_T^+ = W_T^- = 1$ and $W_F^+ = W_F^- = 0$, that is, we seek to maximise the classification accuracy regardless of the dataset imbalance level. To evaluate both the formulation introduced in Section 3.1 and the diversity constraints introduced in Section 3.2, we propose three different configurations.

In the first we employ only constraints (3.2)-(3.12), without enforcing diversity constraints. We refer to this configuration as F1. The other two configurations, F2 and F3, enforce minimum diversity levels in the hope of preventing possible overfitting of the training and validation sets. In F2 we set $\tau = 0$ and $\gamma = \frac{\text{PFC}_{\text{min}} + \text{PFC}_{\text{avg}}}{2}$, where PFC_{min} and PFC_{avg} are the minimum individual PFC and the average PFC value of the full ensemble. So in F2 we only constrain the overall average PFC value. In F3 we also set $\tau = \text{PFC}_{\text{min}}$, so restricting individual PFC values as well as the average PFC. Here the F1 and F3 configurations are equivalent to the Section 4.2.

Table 4.4: Out-of-sample average accuraries and standard deviations.

Dataset	K	F1		F2		F3		Kappa pruning		Backfitting		Full Ensemble	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
PRK	40	**0.90660**	0.01453	**0.91340**	0.01364	**0.91140**	0.01297	0.89071	0.01487	0.90407	0.01556	0.90143	0.00656
	60	**0.90360**	0.01236	**0.90890**	0.01025	**0.90837**	0.01134	0.88018	0.00858	0.89988	0.01457	0.89937	0.00876
	80	**0.90512**	0.01436	**0.90518**	0.01245	**0.90565**	0.01306	0.88374	0.01265	0.90151	0.00838	0.90040	0.00909
	100	**0.90362**	0.01535	**0.90704**	0.02023	**0.90415**	0.01828	0.88032	0.00912	0.90199	0.00804	0.89585	0.00925
BCW	40	**0.96637**	0.00511	**0.96887**	0.00522	**0.96904**	0.00536	0.96079	0.00675	0.96517	0.00580	0.96324	0.00578
	60	**0.96762**	0.00431	**0.97008**	0.00359	**0.97026**	0.00382	0.96184	0.00535	0.96710	0.00558	0.96288	0.00559
	80	0.96426	0.00573	**0.96729**	0.00324	**0.96729**	0.00324	0.96287	0.00292	0.96603	0.00485	0.96287	0.00507
	100	0.96180	0.00523	**0.96764**	0.00355	**0.96764**	0.00400	0.96358	0.00392	0.96727	0.00469	0.96270	0.00565
MSK	40	0.91086	0.01098	**0.91651**	0.00928	**0.91272**	0.00901	0.89231	0.00884	0.91166	0.01218	0.88961	0.01005
	60	**0.91189**	0.00855	**0.91294**	0.00917	**0.91147**	0.00911	0.88810	0.01284	0.90831	0.01361	0.88687	0.00967
	80	**0.91917**	0.00973	**0.92084**	0.00779	**0.92084**	0.00759	0.89460	0.01399	0.91628	0.01165	0.88938	0.00994
	100	**0.91965**	0.00658	**0.91920**	0.00748	**0.91878**	0.00849	0.89062	0.01216	0.90976	0.01347	0.88854	0.00958
QSR	40	**0.87395**	0.00406	0.87308	0.00379	**0.87423**	0.00469	0.87345	0.00441	0.87194	0.00462	0.87062	0.00327
	60	**0.87546**	0.00562	**0.87555**	0.00452	**0.87403**	0.00550	0.87080	0.00482	0.87101	0.00502	0.86920	0.00390
	80	0.87147	0.00475	0.87280	0.00473	0.87403	0.00329	0.87137	0.00380	0.87469	0.00449	0.87062	0.00250
	100	0.87223	0.00541	**0.87565**	0.00267	0.87318	0.00419	0.87385	0.00532	0.87289	0.00502	0.87052	0.00281
DRD	40	0.74007	0.00598	0.74050	0.00910	0.73885	0.00747	0.73128	0.00848	0.74145	0.00795	0.70862	0.00801
	60	0.74579	0.01137	0.74622	0.00964	0.74475	0.01038	0.73675	0.00930	0.74753	0.01002	0.71105	0.00766
	80	0.74493	0.00627	0.74118	0.00539	0.74371	0.00683	0.71340	0.01084	0.74641	0.00599	0.70940	0.00785
	100	0.74006	0.00834	**0.74588**	0.00539	**0.74424**	0.00651	0.71436	0.01386	0.74415	0.00645	0.71140	0.00764
SPA	40	0.95286	0.00144	**0.95344**	0.00128	**0.95344**	0.00131	0.94554	0.00110	0.95286	0.00080	0.94762	0.00104
	60	0.95331	0.00133	**0.95423**	0.00150	**0.95371**	0.00172	0.94475	0.00106	0.95347	0.00145	0.94695	0.00070
	80	**0.95379**	0.00180	**0.95362**	0.00203	**0.95362**	0.00143	0.94501	0.00104	0.95318	0.00137	0.94677	0.00079
	100	**0.95353**	0.00182	**0.95384**	0.00157	**0.95371**	0.00162	0.94456	0.00108	0.95299	0.00135	0.94669	0.00065
DEF	40	0.82061	0.00050	0.82047	0.00052	0.82041	0.00049	0.82057	0.00052	0.82064	0.00039	0.82067	0.00042
	60	0.82048	0.00050	0.82035	0.00053	0.82057	0.00029	0.82093	0.00028	0.82068	0.00029	0.82048	0.00055
	80	**0.82033**	0.00056	**0.82064**	0.00061	**0.82063**	0.00077	0.82014	0.00076	0.82031	0.00037	0.82030	0.00041
	100	0.82019	0.00039	0.82005	0.00033	0.82052	0.00057	0.81976	0.00103	0.82061	0.00043	0.82010	0.00040
BMK	40	**0.91781**	0.00045	**0.91712**	0.00062	**0.91701**	0.00061	0.91322	0.00047	0.91661	0.00059	0.91435	0.00074
	60	**0.91749**	0.00049	**0.91731**	0.00058	**0.91734**	0.00062	0.91167	0.00040	0.91668	0.00035	0.91341	0.00071
	80	**0.91743**	0.00070	**0.91745**	0.00066	**0.91720**	0.00051	0.91094	0.00049	0.91647	0.00056	0.91306	0.00052
	100	**0.91757**	0.00036	**0.91740**	0.00052	**0.91725**	0.00069	0.91005	0.00043	0.91669	0.00042	0.91242	0.00056
APS	40	**0.99369**	0.00021	**0.99376**	0.00016	**0.99376**	0.00016	0.99329	0.00016	0.99361	0.00021	0.99287	0.00014
	60	**0.99377**	0.00018	**0.99378**	0.00016	**0.99381**	0.00019	0.99293	0.00018	0.99357	0.00021	0.99250	0.00013
	80	**0.99385**	0.00010	**0.99380**	0.00013	**0.99382**	0.00011	0.99251	0.00015	0.99357	0.00017	0.99254	0.00013
	100	**0.99382**	0.00015	**0.99389**	0.00014	**0.99391**	0.00012	0.99242	0.00014	0.99369	0.00021	0.99247	0.00013
Average:		**0.89847**	**0.00488**	**0.89972**	**0.00451**	**0.89932**	**0.00462**	**0.88926**	**0.00506**	**0.89791**	**0.00492**	**0.88938**	**0.00407**

Table 4.4 presents the results for the 9 datasets and four different values of K. For each algorithm, we include two columns: the average (Avg.) and standard deviation

(Std.) of the out-of-sample accuracy, calculated with the data points in the independent set. We remind the reader that each entry in the table represents the average accuracy of 100 different runs.

A bold value in any of the first three Avg. columns means that our formulation obtained a higher average than all three benchmarks. The last row gives an overall average value of the corresponding column.

Backfitting, which explicitly attempts to maximise accuracy, had a higher overall average accuracy than the other benchmarks. F1 obtained slightly better results when compared to Backfitting: an overall average accuracy of 0.89847 compared to 0.89791 and a very slightly lower standard deviation (0.00488 as opposed to 0.00492). In 22 out of the 36 cases, F1 obtained a higher average accuracy than all three benchmarks.

Results for F2 and F3 are improved when compared to F1. The overall average accuracy increased to 0.89972 for F2 and 0.89932 for F3. The overall average accuracy standard deviation decreased to 0.00451 and 0.00462. Both configurations were able to obtain higher average accuracy for 28 out of the 36 cases. F2 and F3 had little discernible difference.

This empirical evidence enforces the common knowledge that both performance and diversity are important when pruning ensembles. It also suggests, in our view, that the flexibility of our IP approach can be explored for improved predictive performance.

The use of integer programming, which attempts to find optimal solutions, may overfit the selected ensemble to the dataset used in the optimisation. To have an overall idea of overfitting, we compare the average accuracy in out-of-sample test data from Table 4.4 with the accuracy of the same selected classifiers, but in the in-sample validation data used in the optimisation approach.

The comparison is in Table 4.5. In the Avg. column, we perform this comparison by calculating the (average out-of-sample accuracy - average in-sample accuracy) / (average in-sample accuracy). As the results are averages of 100 executions, we also include the standard deviation in column **Std.**. The last row includes the average of all datasets and values of K from its respective column. The more negative the values are, the more the accuracy of the out-of-sample decreased related to the in-sample, which may be a sign of overfitting to the in-sample data.

In the last row, Kappa Pruning has an average difference of -0.00453, which is the best average among the results, followed by Backfitting with an average of -0.02528. The average for F1, F2 and F3 are -0.03304, -0.03127 and -0.03143, respectively. They are more negative than the corresponding values for Kappa Pruning and Backfitting.

From the table we can see that the overfitting problem affected the formulations more strongly than the benchmark algorithms. We consider this as expected behaviour since the formulation is solved by seeking optimality, while the other algorithms offer no guarantees of optimality. As such, the accuracies of the formulations tend to be higher than the Kappa Pruning and Backfitting algorithms in the in-sample sets, but deteriorate more out-of-sample. However, even with higher deterioration, Table 4.4 suggests that the out-of-sample accuracies of the formulations were similar or superior to the benchmarks. This may indicate that with better overfitting control we may be able to improve the out-of-sample results of our formulation.

Table 4.5: Comparation between in-sample and out-of-sample accuracies

Dataset	K	**F1**		**F2**		**F3**		Kappa Pruning		Backfitting	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
PRK	40	-0.05825	0.02230	-0.05125	0.01649	-0.05335	0.01727	-0.01857	0.02397	-0.05286	0.02566
	60	-0.06580	0.01793	-0.06039	0.01433	-0.06087	0.01704	-0.02684	0.02654	-0.05977	0.02635
	80	-0.06671	0.02418	-0.06661	0.02204	-0.06614	0.02142	-0.02476	0.02718	-0.06109	0.01778
	100	-0.06989	0.02372	-0.06633	0.02832	-0.06933	0.02699	-0.02330	0.02776	-0.05748	0.01657
BCW	40	-0.02440	0.00573	-0.02188	0.00623	-0.02171	0.00651	-0.00433	0.00671	-0.01899	0.00835
	60	-0.02474	0.00501	-0.02224	0.00507	-0.02206	0.00533	-0.00212	0.00454	-0.01812	0.00942
	80	-0.02915	0.00682	-0.02608	0.00455	-0.02608	0.00455	-0.00407	0.00509	-0.01955	0.00742
	100	-0.03233	0.00628	-0.02644	0.00443	-0.02644	0.00483	-0.00409	0.00646	-0.01880	0.00704
MSK	40	-0.05032	0.01493	-0.04446	0.01157	-0.04842	0.01215	0.00450	0.02122	-0.03436	0.01423
	60	-0.05183	0.01561	-0.05079	0.01313	-0.05230	0.01417	-0.00032	0.02249	-0.03842	0.01665
	80	-0.05218	0.01515	-0.05052	0.01305	-0.05049	0.01309	-0.00266	0.02359	-0.03517	0.01844
	100	-0.05293	0.01083	-0.05347	0.01081	-0.05386	0.01375	-0.00384	0.02169	-0.04038	0.01870
QSR	40	-0.03809	0.00662	-0.03902	0.00744	-0.03773	0.00892	-0.00369	0.01032	-0.03155	0.01076
	60	-0.04133	0.00876	-0.04124	0.00755	-0.04291	0.00794	-0.00245	0.00897	-0.03414	0.01068
	80	-0.04993	0.01001	-0.04845	0.01081	-0.04710	0.00913	-0.00384	0.01035	-0.03317	0.00784
	100	-0.05188	0.00876	-0.04811	0.00713	-0.05076	0.00936	-0.00015	0.01184	-0.03560	0.00762
DRD	40	-0.07302	0.00940	-0.07190	0.01374	-0.07418	0.01309	-0.00829	0.01688	-0.05161	0.01237
	60	-0.07864	0.01534	-0.07657	0.01223	-0.07862	0.01240	-0.00580	0.01449	-0.05563	0.01300
	80	-0.08580	0.01184	-0.08662	0.00807	-0.08070	0.01029	-0.01033	0.01443	-0.06383	0.00871
	100	-0.09828	0.01143	-0.08438	0.01031	-0.08034	0.01159	-0.01178	0.01902	-0.07141	0.00936
SPA	40	-0.01183	0.00300	-0.01122	0.00233	-0.01122	0.00241	-0.00244	0.00193	-0.00923	0.00242
	60	-0.01303	0.00244	-0.01207	0.00248	-0.01262	0.00325	-0.00169	0.00225	-0.01040	0.00264
	80	-0.01357	0.00245	-0.01374	0.00320	-0.01374	0.00243	-0.00158	0.00226	-0.01135	0.00148
	100	-0.01465	0.00248	-0.01418	0.00194	-0.01431	0.00198	-0.00173	0.00204	-0.01196	0.00193
DEF	40	-0.00459	0.00158	-0.00449	0.00124	-0.00449	0.00140	-0.00004	0.00094	-0.00352	0.00137
	60	-0.00516	0.00158	-0.00493	0.00152	-0.00448	0.00141	0.00036	0.00107	-0.00386	0.00100
	80	-0.00541	0.00145	-0.00475	0.00191	-0.00457	0.00123	0.00022	0.00136	-0.00470	0.00098
	100	-0.00568	0.00152	-0.00512	0.00156	-0.00423	0.00171	0.00037	0.00127	-0.00448	0.00130
BMK	40	-0.00316	0.00093	-0.00350	0.00099	-0.00356	0.00124	-0.00006	0.00109	-0.00324	0.00111
	60	-0.00375	0.00091	-0.00354	0.00102	-0.00341	0.00117	-0.00002	0.00113	-0.00354	0.00101
	80	-0.00407	0.00125	-0.00342	0.00095	-0.00345	0.00108	-0.00001	0.00097	-0.00421	0.00121
	100	-0.00396	0.00090	-0.00319	0.00089	-0.00315	0.00134	0.00007	0.00115	-0.00432	0.00075
APS	40	-0.00118	0.00026	-0.00110	0.00025	-0.00109	0.00024	-0.00002	0.00021	-0.00069	0.00017
	60	-0.00122	0.00024	-0.00119	0.00022	-0.00116	0.00027	0.00002	0.00023	-0.00076	0.00016
	80	-0.00131	0.00017	-0.00132	0.00016	-0.00129	0.00015	0.00002	0.00021	-0.00091	0.00019
	100	-0.00145	0.00018	-0.00131	0.00014	-0.00126	0.00016	-0.00001	0.00021	-0.00093	0.00022
Average:		**-0.03304**	**0.00756**	**-0.03127**	**0.00689**	**-0.03143**	**0.00726**	**-0.00453**	**0.00950**	**-0.02528**	**0.00791**

4.3.3 Area Under the Curve

In this section, we employ the out-of-sample AUC instead of accuracy when we comparing the formulation and the benchmarks. For that we use two different assignments of weights to objective function (3.1). Here the goal is to get insights on how seemingly more appropriate weights impact the out-of-sample AUC performance, better suited for imbalanced datasets.

As explained in the beginning of the Chapter 4, we round the probabilities given by each classifier to each data point in order to generate A and B. AUC however requires probability based predictions instead of a binary classifications. After employing our proposed formulation, we ignore the threshold L given and for each data point we take the average probabilistic prediction for the ensemble subset, so also ignoring the rounding used to create A and B. AUC is then calculated by varying a probability threshold between 0 and 1. AUC is generally considered to be more informative than accuracy for imbalanced data [16].

We use two weights assignments: maximising accuracy and maximising the θ-weighted configuration suggested in Section 3.1.1. The latter does not exactly maximise AUC but, by taking into account class imbalance, may be more suited for improving this particular performance metric. Here we do not apply any diversity constraints.

Table 4.6: Out-of-sample average AUC's and standard deviations.

Dataset	K	F1 (Accuracy)		F1 (θ-weighted)		Kappa Pruning		Backfitting		Full Ensemble	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
PRK	40	**0.95719**	0.00900	0.95553	0.01005	0.94626	0.01018	0.95358	0.01292	0.95701	0.00707
	60	0.95441	0.01127	**0.95514**	0.00945	0.94168	0.01183	0.95274	0.01413	0.95449	0.00630
	80	0.94861	0.01782	0.95269	0.01589	0.94352	0.01396	0.95418	0.01061	0.95690	0.00477
	100	0.95239	0.01487	**0.95640**	0.01148	0.93909	0.01174	0.95492	0.01106	0.95436	0.00657
BCW	40	0.99426	0.00137	**0.99489**	0.00096	0.99425	0.00094	0.99423	0.00134	0.99458	0.00103
	60	0.99457	0.00132	**0.99516**	0.00067	0.99431	0.00133	0.99475	0.00068	0.99449	0.00089
	80	0.99466	0.00122	0.99477	0.00123	0.99377	0.00109	0.99484	0.00088	0.99449	0.00094
	100	0.99520	0.00088	0.99498	0.00104	0.99415	0.00120	0.99524	0.00100	0.99463	0.00087
MSK	40	**0.97033**	0.00465	**0.96959**	0.00496	0.95494	0.00686	0.96845	0.00491	0.95817	0.00422
	60	0.96963	0.00418	**0.96980**	0.00647	0.95526	0.00637	0.96530	0.00541	0.95549	0.00436
	80	**0.97484**	0.00525	**0.97312**	0.00542	0.95903	0.00661	0.97092	0.00427	0.95795	0.00402
	100	**0.97487**	0.00479	**0.97459**	0.00508	0.95752	0.00612	0.96872	0.00450	0.95636	0.00400
QSR	40	0.93526	0.00221	**0.93660**	0.00261	0.93574	0.00158	0.93495	0.00251	0.93624	0.00208
	60	0.93562	0.00218	0.93571	0.00227	0.93591	0.00232	0.93653	0.00199	0.93559	0.00204
	80	0.93547	0.00331	0.93607	0.00191	0.93608	0.00210	0.93670	0.00228	0.93606	0.00196
	100	0.93675	0.00302	0.93661	0.00128	0.93605	0.00188	0.93688	0.00289	0.93592	0.00202
DRD	40	**0.82866**	0.00522	**0.82785**	0.00436	0.81892	0.00470	0.82692	0.00606	0.79653	0.00459
	60	**0.82979**	0.00410	0.82815	0.00297	0.82041	0.00606	0.82931	0.00578	0.79664	0.00413
	80	0.82995	0.00437	0.82930	0.00425	0.80570	0.00690	0.83053	0.00680	0.79580	0.00426
	100	**0.82710**	0.00429	**0.82950**	0.00480	0.80639	0.00699	0.82702	0.00670	0.79777	0.00430
SPA	40	0.98819	0.00037	0.98827	0.00036	0.98730	0.00029	0.98830	0.00065	0.98804	0.00023
	60	0.98831	0.00067	0.98807	0.00038	0.98651	0.00035	0.98843	0.00040	0.98748	0.00025
	80	**0.98834**	0.00062	**0.98833**	0.00050	0.98678	0.00028	0.98818	0.00056	0.98771	0.00026
	100	0.98836	0.00060	0.98843	0.00064	0.98658	0.00029	0.98854	0.00055	0.98760	0.00024
DEF	40	0.78162	0.00167	0.78337	0.00057	0.78357	0.00063	0.78183	0.00084	0.78380	0.00043
	60	0.78069	0.00204	**0.78359**	0.00049	0.78352	0.00059	0.78121	0.00097	0.78318	0.00036
	80	0.78006	0.00218	**0.78355**	0.00055	0.78227	0.00169	0.78032	0.00134	0.78331	0.00055
	100	0.77838	0.00275	**0.78317**	0.00058	0.78177	0.00160	0.77997	0.00131	0.78278	0.00045
BMK	40	**0.94862**	0.00046	0.94827	0.00027	0.94692	0.00024	0.94833	0.00060	0.94809	0.00026
	60	0.94843	0.00021	0.94813	0.00045	0.94600	0.00028	0.94848	0.00028	0.94744	0.00023
	80	0.94836	0.00044	0.94808	0.00025	0.94609	0.00025	0.94863	0.00024	0.94750	0.00023
	100	0.94835	0.00031	0.94817	0.00040	0.94549	0.00027	0.94848	0.00029	0.94716	0.00022
APS	40	0.98322	0.00257	0.98280	0.00305	0.97636	0.00195	0.98634	0.00190	0.98359	0.00180
	60	0.98147	0.00309	0.98286	0.00234	0.97914	0.00179	0.98532	0.00240	0.98237	0.00134
	80	0.97948	0.00359	**0.98384**	0.00205	0.98216	0.00188	0.98268	0.00308	0.98357	0.00177
	100	0.98037	0.00317	0.98387	0.00193	0.98276	0.00201	0.98176	0.00292	0.98387	0.00175
Average:		**0.93144**	**0.00361**	**0.93220**	**0.00311**	**0.92645**	**0.00348**	**0.93149**	**0.00347**	**0.92686**	**0.00224**

Table 4.6, which has the same structure as Table 4.4, presents the results. We

did not rerun the experiments for the accuracy version of F1 nor for the benchmarks, rather we used the same ensemble subsets to calculate the corresponding AUC's. Once again, each entry in the Avg. column represents the average AUC of 100 executions, with corresponding standard deviations given in column Std..

Backfitting still had the best overall average AUC among the benchmarks (0.93149 with 0.0347 standard deviation). Its performance was very similar to F1 with accuracy. However, by changing the weights assignment to the θ-weighted configuration, we were able to considerably improve the results. The overall average AUC increased to 0.93220 with a slightly reduced overall standard deviation of 0.00311. In 9 out of 36 cases, F1 with accuracy maximisation obtained a higher average AUC than all benchmarks. This number increased to 16 when using the θ-weighted configuration.

4.3.4 Balanced accuracy

In this section, we analyse the same assignments of weights from last section to (3.1), that is, maximising accuracy and maximising the θ-weighted configuration suggested in Section 3.1.1. However here we evaluate a different performance metric, namely the balanced accuracy. AUC is generally considered as a suitable metric for imbalanced datasets, however it requires predicted probabilities to be calculated, while often in real world problems we need to use 0 or 1 classifications instead of probabilities. This way, we decided to also evaluate our results with a metric that is more related to the context of a binary classification problem. The balanced accuracy is equal to the accuracy of the out-of-sample data points of the positive class plus the accuracy of the out-of-sample data points of the negative class divided by 2.

Table 4.7, which has the same structure as Table 4.4, presents the results. We used the same ensemble subsets from the last section to calculate the corresponding balanced accuracy. Once again, each entry in the Avg. column represents the average AUC of 100 executions, with corresponding standard deviations given in column Std..

Backfitting once again had the best overall metric results among the benchmarks (0.83128 with 0.00689 standard deviation). The use of the weights assignment to the θ-weighted configuration has considerably improved the results of balanced accuracy. In 31 out of 36 cases, F1 with accuracy maximisation obtained a higher average balanced accuracy than all benchmarks. This number increased to 32 when using the θ-weighted configuration. The more noticeable increase, however, can be seen in the overall average balanced accuracy: it increased from 0.84332 using the F1 original configuration to 0.86645 when using the θ-weighted configuration, and the standard deviation decreased

Table 4.7: Out-of-sample average balanced accuraries and standard deviations.

Dataset	K	F1		F1 (θ-weighted)		Kappa Pruning		Backfitting		Full Ensemble	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
PRK	40	0.84781	0.03275	**0.85821**	0.02893	0.80785	0.02071	0.85230	0.01768	0.83083	0.01097
	60	**0.84668**	0.02422	**0.85446**	0.01805	0.77599	0.01359	0.83643	0.02422	0.82374	0.01575
	80	**0.84635**	0.02897	**0.85144**	0.02045	0.78475	0.01961	0.84326	0.01694	0.82640	0.01461
	100	**0.84867**	0.03063	**0.85565**	0.02664	0.77582	0.01500	0.84160	0.01035	0.81540	0.01199
BCW	40	**0.96449**	0.00566	**0.96682**	0.00496	0.95428	0.00792	0.95814	0.00660	0.95659	0.00773
	60	**0.96663**	0.00520	**0.96754**	0.00371	0.95560	0.00653	0.96082	0.00637	0.95641	0.00726
	80	**0.96455**	0.00545	**0.96507**	0.00444	0.95639	0.00385	0.96006	0.00588	0.95621	0.00689
	100	**0.96249**	0.00505	**0.96315**	0.00563	0.95745	0.00474	0.96162	0.00590	0.95617	0.00732
MSK	40	**0.91027**	0.01022	**0.91039**	0.00826	0.88718	0.01034	0.90714	0.01332	0.88436	0.01080
	60	**0.91084**	0.00797	**0.90942**	0.00838	0.88344	0.01418	0.90466	0.01429	0.88146	0.01020
	80	**0.91787**	0.01005	**0.91642**	0.00747	0.88993	0.01496	0.91211	0.01241	0.88408	0.01046
	100	**0.91798**	0.00712	**0.91616**	0.00577	0.88585	0.01305	0.90559	0.01433	0.88329	0.01022
QSR	40	**0.85742**	0.00656	**0.86445**	0.00494	0.84888	0.00524	0.84963	0.00520	0.84318	0.00365
	60	**0.86042**	0.00719	**0.86386**	0.00524	0.84324	0.00559	0.84668	0.00511	0.84072	0.00396
	80	**0.85588**	0.00456	**0.86492**	0.00379	0.84498	0.00587	0.85149	0.00498	0.84206	0.00251
	100	**0.85632**	0.00498	**0.86481**	0.00523	0.84777	0.00669	0.84772	0.00490	0.84186	0.00297
DRD	40	0.74071	0.00561	0.74340	0.00617	0.73642	0.00866	0.74651	0.00782	0.71169	0.00793
	60	0.74631	0.01145	0.74370	0.00884	0.74160	0.00925	0.75199	0.00983	0.71452	0.00750
	80	0.74568	0.00639	0.74679	0.00455	0.71755	0.01046	0.75033	0.00572	0.71255	0.00773
	100	0.74039	0.00881	0.74664	0.00932	0.71832	0.01331	0.74791	0.00631	0.71479	0.00750
SPA	40	**0.94929**	0.00176	**0.95128**	0.00226	0.93844	0.00143	0.94779	0.00120	0.94151	0.00127
	60	**0.94981**	0.00165	**0.95144**	0.00199	0.93756	0.00134	0.94895	0.00179	0.94053	0.00082
	80	**0.95039**	0.00212	**0.95210**	0.00184	0.93783	0.00120	0.94881	0.00174	0.94033	0.00089
	100	**0.94996**	0.00223	**0.95171**	0.00249	0.93722	0.00124	0.94847	0.00196	0.94015	0.00078
DEF	40	**0.66335**	0.00218	**0.69488**	0.00228	0.65112	0.00157	0.65069	0.00108	0.65067	0.00112
	60	**0.66428**	0.00204	**0.69731**	0.00214	0.65099	0.00091	0.64914	0.00184	0.64898	0.00120
	80	**0.66616**	0.00139	**0.69851**	0.00163	0.64895	0.00288	0.64845	0.00197	0.64840	0.00111
	100	**0.66607**	0.00188	**0.69921**	0.00141	0.64680	0.00413	0.64819	0.00181	0.64730	0.00123
BMK	40	**0.77307**	0.00548	**0.86071**	0.00152	0.68622	0.00271	0.73307	0.00376	0.69926	0.00388
	60	**0.77631**	0.00488	**0.86616**	0.00186	0.66847	0.00329	0.73375	0.00281	0.68829	0.00376
	80	**0.77655**	0.00552	**0.86844**	0.00155	0.65940	0.00369	0.73221	0.00462	0.68219	0.00363
	100	**0.77941**	0.00527	**0.86939**	0.00122	0.65221	0.00326	0.73631	0.00404	0.67618	0.00314
APS	40	**0.86904**	0.00621	**0.93815**	0.00402	0.82961	0.00644	0.84039	0.00574	0.81550	0.00371
	60	**0.87154**	0.00617	**0.93860**	0.00369	0.81498	0.00697	0.83920	0.00471	0.80149	0.00331
	80	**0.87306**	0.00391	**0.93947**	0.00380	0.80086	0.00525	0.83984	0.00453	0.80205	0.00371
	100	**0.87348**	0.00532	**0.94160**	0.00412	0.79968	0.00549	0.84472	0.00640	0.80064	0.00324
Average:		**0.84332**	**0.00797**	**0.86645**	**0.00635**	**0.80760**	**0.00726**	**0.83128**	**0.00689**	**0.81111**	**0.00569**

from 0.00797 to 0.00635. We see this as empirical evidence that the flexibility of setting problem-dependent weights is effective when we need to give more importance to one class over the other. Furthermore, other empirical evidence that corroborate our affirmative are the AUC metric results from Section 4.3.3, where the θ-weighted configuration was also able to perform better compared to F1 and the benchmarks.

4.4 Majority Voting Formulation

In a final experiment we tested the use of majority voting to define the classification threshold on our formulations. In the formulations of the Chapter 3 we left the threshold variable L free to get any value in $0 < L < S$, where S is the number of classifiers selected in the ensemble. In order to use the majority voting, we have to ensure that the value of L is equal to $\lfloor S/2 \rfloor$. Constraint (4.1) can be added to the formulation to ensure this.

$$\frac{1}{2}\sum_{k=1}^{K} x_k - 0.5 \leq L \leq \frac{1}{2}\sum_{k=1}^{K} x_k \tag{4.1}$$

The use of the majority voting may also help to reduce overfitting, since the optimisation will use only a subset of the thresholds values in comparison with the original formulation.

In our experiments, we repeated the configurations of F1, F2 and F3 from Section 4.3.2, but adding the constraints 4.1 to ensure the use of majority voting. The new configurations are respectively refered as FMV1, FMV2 and FMV3.

Table 4.8: Out-of-sample average accuraries and standard deviations for Majority Voting Formulations and Benchmarks.

Dataset	K	FMV1		FMV2		FMV3		F2		Kappa Pruning		Backfitting		Full Ensemble	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
PRK	40	**0.91284**	0.01535	**0.91340**	0.01022	**0.91120**	0.01182	**0.91340**	0.01364	0.89071	0.01487	0.90407	0.01556	0.90143	0.00656
	60	**0.91040**	0.01531	**0.91095**	0.01232	**0.91098**	0.01173	**0.90890**	0.01025	0.88018	0.00858	0.89988	0.01457	0.89937	0.00876
	80	**0.90868**	0.01330	**0.91184**	0.01183	**0.91026**	0.01403	**0.90518**	0.01245	0.88374	0.01265	0.90151	0.00838	0.90040	0.00909
	100	**0.90806**	0.01338	**0.90907**	0.01431	**0.90976**	0.01709	**0.90704**	0.02023	0.88032	0.00912	0.90199	0.00804	0.89585	0.00925
BCW	40	**0.96939**	0.00662	**0.97098**	0.00685	**0.97012**	0.00574	**0.96887**	0.00522	0.96079	0.00675	0.96517	0.00580	0.96324	0.00578
	60	**0.97028**	0.00683	**0.97097**	0.00595	**0.97098**	0.00623	**0.97008**	0.00359	0.96184	0.00535	0.96710	0.00558	0.96288	0.00559
	80	**0.96763**	0.00592	**0.96834**	0.00491	**0.96799**	0.00547	**0.96729**	0.00324	0.96287	0.00292	0.96603	0.00485	0.96287	0.00507
	100	0.96656	0.00538	**0.97025**	0.00442	**0.96954**	0.00582	**0.96764**	0.00355	0.96358	0.00392	0.96727	0.00469	0.96270	0.00565
MSK	40	**0.91604**	0.01211	**0.91753**	0.01187	**0.91732**	0.01437	**0.91651**	0.00928	0.89231	0.00884	0.91166	0.01218	0.88961	0.01005
	60	**0.91880**	0.00912	**0.91776**	0.01223	**0.91901**	0.01027	**0.91294**	0.00917	0.88810	0.01284	0.90831	0.01361	0.88687	0.00967
	80	**0.92174**	0.01164	**0.92131**	0.00994	**0.92109**	0.01091	**0.92084**	0.00779	0.89460	0.01399	0.91628	0.01165	0.88938	0.00994
	100	**0.91652**	0.00753	**0.91882**	0.00842	**0.91987**	0.00991	**0.91920**	0.00748	0.89062	0.01216	0.90976	0.01347	0.88854	0.00958
QSR	40	**0.87346**	0.00463	**0.87574**	0.00429	**0.87498**	0.00517	0.87308	0.00379	0.87345	0.00441	0.87194	0.00462	0.87062	0.00327
	60	**0.87346**	0.00593	**0.87422**	0.00491	**0.87384**	0.00488	**0.87555**	0.00452	0.87080	0.00482	0.87101	0.00502	0.86920	0.00390
	80	0.87290	0.00404	0.87403	0.00623	0.87374	0.00359	0.87280	0.00473	0.87137	0.00380	0.87469	0.00449	0.87062	0.00250
	100	**0.87432**	0.00336	**0.87658**	0.00284	**0.87404**	0.00377	**0.87565**	0.00267	0.87385	0.00532	0.87289	0.00502	0.87052	0.00281
DRD	40	**0.74475**	0.00903	**0.74449**	0.00725	**0.74293**	0.00920	0.74050	0.00910	0.73128	0.00848	0.74145	0.00795	0.70862	0.00801
	60	**0.74865**	0.00673	0.74719	0.00805	**0.74840**	0.00875	0.74622	0.00964	0.73675	0.00930	0.74753	0.01002	0.71105	0.00766
	80	**0.74849**	0.00652	0.74485	0.00826	0.74484	0.00527	0.74118	0.00539	0.71340	0.01084	0.74641	0.00599	0.70940	0.00785
	100	**0.74891**	0.00922	0.74362	0.00638	0.74353	0.00690	**0.74588**	0.00539	0.71436	0.01386	0.74415	0.00645	0.71140	0.00764
SPA	40	0.95260	0.00118	0.95262	0.00153	**0.95316**	0.00163	**0.95344**	0.00128	0.94554	0.00110	0.95286	0.00080	0.94762	0.00104
	60	**0.95364**	0.00122	**0.95384**	0.00090	**0.95408**	0.00148	**0.95423**	0.00150	0.94475	0.00106	0.95347	0.00145	0.94695	0.00070
	80	**0.95349**	0.00141	**0.95442**	0.00211	**0.95373**	0.00184	**0.95362**	0.00203	0.94501	0.00104	0.95318	0.00137	0.94677	0.00079
	100	**0.95353**	0.00138	**0.95379**	0.00089	**0.95358**	0.00125	**0.95384**	0.00157	0.94456	0.00108	0.95299	0.00135	0.94669	0.00065
DEF	40	0.82057	0.00046	0.82044	0.00045	0.82040	0.00043	0.82047	0.00052	0.82057	0.00052	0.82064	0.00039	0.82067	0.00042
	60	0.82050	0.00027	0.82044	0.00050	0.82069	0.00045	0.82035	0.00053	0.82093	0.00028	0.82068	0.00029	0.82048	0.00055
	80	**0.82038**	0.00052	**0.82079**	0.00037	**0.82036**	0.00036	**0.82064**	0.00061	0.82014	0.00076	0.82031	0.00037	0.82030	0.00041
	100	**0.82073**	0.00042	**0.82068**	0.00035	**0.82066**	0.00042	0.82005	0.00033	0.81976	0.00103	0.82061	0.00043	0.82010	0.00040
BMK	40	**0.91702**	0.00055	**0.91679**	0.00059	**0.91675**	0.00068	**0.91712**	0.00062	0.91322	0.00047	0.91661	0.00059	0.91435	0.00074
	60	**0.91696**	0.00036	**0.91696**	0.00073	0.91664	0.00048	**0.91731**	0.00058	0.91167	0.00040	0.91668	0.00035	0.91341	0.00071
	80	**0.91719**	0.00039	**0.91696**	0.00035	**0.91668**	0.00060	**0.91745**	0.00066	0.91094	0.00049	0.91647	0.00056	0.91306	0.00052
	100	**0.91706**	0.00040	**0.91701**	0.00060	**0.91694**	0.00069	**0.91740**	0.00052	0.91005	0.00043	0.91669	0.00042	0.91242	0.00056
APS	40	**0.99368**	0.00020	**0.99373**	0.00018	**0.99367**	0.00017	**0.99376**	0.00016	0.99328	0.00016	0.99361	0.00021	0.99288	0.00014
	60	**0.99367**	0.00020	**0.99370**	0.00020	**0.99370**	0.00022	**0.99378**	0.00016	0.99293	0.00018	0.99357	0.00021	0.99250	0.00013
	80	**0.99376**	0.00028	**0.99377**	0.00024	**0.99372**	0.00018	**0.99380**	0.00013	0.99251	0.00015	0.99357	0.00017	0.99254	0.00013
	100	**0.99379**	0.00021	**0.99379**	0.00015	**0.99384**	0.00022	**0.99389**	0.00014	0.99242	0.00014	0.99369	0.00021	0.99247	0.00013
Average:		**0.90029**	**0.00504**	**0.90060**	**0.00477**	**0.90036**	**0.00506**	**0.89972**	**0.00451**	**0.88926**	**0.00506**	**0.89791**	**0.00492**	**0.88938**	**0.00407**

Table 4.4 presents the results and has the same structure as the previous results tables. We keep **F2** for comparison. The results for all three configurations are very similar to each other and to F2, showing a small improvement in overall average values and a slight increase in average standard deviations. We believe it is not possible yet to conclude whether it is advantageous to let L to be chosen by the algorithm or to fix it prior to solving the problem.

4.5 Summary

In this chapter we outlined the computational experiments used to evaluate both the solution performance and out-of-sample results of our formulation.

In Section 4.2, when analysing the solution performance, we observed that good solutions for our formulation can be found in reasonable computational times, even for the hardest instances. The inclusion of the diversity constraints caused a negligible decrease in performance.

In the out-of-sample results in Section 4.3, we evaluated our formulation with a Stratified K-Fold Cross Validation procedure using 9 datasets. We compared our formulation with 3 benchmarks: Kappa Pruning, Backfitting and the Full Ensemble, and we evaluated it using the metrics accuracy, AUC and balanced accuracy.

In Subsection 4.3.2, we evaluated the accuracy of the F1, F2 and F3 configurations. All configurations obtained a higher average accuracy than the benchmarks. The use of diversity constraints improved the results of F2 and F3 when compared with F1. We also verified that our formulation is more suitable to overfitting when compared to the benchmarks.

Next, we tested the F1 and F1 (θ-weighted) configurations. In Subsection 4.3.3 we evaluated their AUC, and in Subsection 4.3.4 the balanced accuracy. Only F1 (θ-weighted) obtained a higher average AUC than the benchmarks. All of them had a higher balanced accuracy when compared to the benchmarks, where the most noticeable results came with F1 (θ-weighted).

Finally, in Subsection 4.4, we tested the use of majority voting to define the classification threshold on the formulations F1, F2 and F3, creating the configurations FMV1, FMV2 and FMV3. By analysing the results, it was not possible to conclude if the use of majority voting constraints is advantageous or not.

Chapter 5

Ongoing research: cutting-planes reformulations

A formulation generally referred to as “cutting-planes” has an exponential number of constraints and thus cannot be readily solved by commercial solvers. To solve it the exponential set of constraints is relaxed (that is, removed from the model) and during the branch-and-bound search we add them on-the-fly, as needed. To add them during the optimisation we need to pause the solver at each node of the branch-and-bound tree, right after the current linear relaxation is solved, and attempt to identify constraints that were relaxed but are violated by the current linear relaxation solution. If any are found we add them to the model and re-solve the current node. We continue this procedure until no other violated constraints are found, moving on to the next node. This method is a simplified description of the Branch-and-cut algorithm [35] and the algorithm that identifies violated constraints based on the current linear relaxation is called a separation algorithm.

Cutting-plane based formulations are often used as a tool in improving computational performance of IP models. The reasoning is that often cutting-planes reformulations have stronger linear relaxation bounds (that is, closer to the optimal integer solution) and that we can solve the original model without necessarily adding all originally removed constraints, or in other words optimality can be proved long before all constraints originally removed are reincluded in the relaxed model. For instance, the most successful exact algorithm for the Travelling-Salesman Problem is a Branch-and-cut [34; 1].

In this section we propose two reformulations in an attempt to tackle a potential drawback of the original formulation. In Constraints (3.2), (3.3), (3.5) and (3.6), a large value of K may for instance weaken the value of the linear relaxation, perhaps making it more difficult for the solvers to prune the branch-and-bound tree and/or prove optimality. The reformulation we propose uses a cutting-planes approach to avoid the use of these constraints.

However, currently the computational experiments for the reformulations are preliminary and further research is necessary to verify whether they are able to improve computational times required to find optimal solutions, especially for larger instances containing more data points and more classifiers. For this reason we introduce the reformulations here but do not present any results, which are left as future work. We do however discuss implementation details and other considerations as this is a topic of ongoing research.

In Section 5.5 we also propose an alternative approach, once again based on cutting planes, to include diversity in the formulation.

5.1 Fixed threshold reformulation

Before presenting this approach we introduce further mathematical notation. Let $\mathcal{K}_i^1 \subseteq \mathcal{K}$ be the set of classifiers which correctly classified data point $i \in \mathcal{N}_1$ as positive. Accordingly let $\mathcal{K}_j^1 \subseteq \mathcal{K}$ be the set of classifiers which incorrectly classified data point $j \in \mathcal{N}_0$ as positive.

In our original formulation, L was defined as a variable classification threshold. In our first proposed cutting-planes we define a fixed threshold $1 \leq D < K$, and thus it must be solved once for every possible value of D. In the next section we propose a similar reformulation which allows for a variable threshold.

For data point $i \in \mathcal{N}_1$, if at least $D+1$ classifiers in $\mathcal{K}_i^1$ are selected to compose the ensemble, then i is correctly classified as positive, or alternatively $t_i^+ = 1$ (true positive). Otherwise we must have that $f_i^- = 1$ (false negative). This behaviour can be captured by employing Constraints (3.4) together and:

$$\sum_{k\in U} x_k - D \leq t_i^+ \qquad \forall i \in \mathcal{N}_1 : |\mathcal{K}_i^1| > D, U \subseteq \mathcal{K}_i^1, |U| = D+1 \tag{5.1}$$

$$1 - \sum_{k\in U} x_k \leq f_i^- \qquad \forall i \in \mathcal{N}_1 : |\mathcal{K}_i^1| > D, U \subseteq \mathcal{K}_i^1, |U| = |\mathcal{K}_i^1| - D \tag{5.2}$$

Constraints (5.1)-(5.2) replace Constraints (3.2)-(3.3). For a given i and D, there is one constraint of type (5.1) for every subset $U \subseteq \mathcal{K}_i^1$ containing exactly $D+1$ classifiers. If enough of these classifiers are selected to make a true positive, then for at least one subset U we have $\sum_{k\in U} x_k = D+1$, which would force $t_i^+ = 1$. For all data points $i \in \mathcal{N}_1$ such that $|\mathcal{K}_i^1| \leq d$, we can fix variables $f_i^- = 1$.

Similarly, if at least $|\mathcal{K}_i^1| - D$ classifiers are not selected to compose the ensemble, then there exists a subset U with size $|\mathcal{K}_i^1| - D$ where $\sum_{k\in U} x_k = 0$, which given the corresponding constraint of type (5.2) would force $f_i^- = 1$.

Note that if $W_T^+ > W_F^-$ Constraints (5.1) are unnecessary as the solver will naturally set $t_i^+ = 1$ if it is possible to set $f_i^- = 0$. Accordingly if $W_T^+ < W_F^-$, we do not need Constraints (5.2). If $W_T^+ = W_F^-$ we need both.

For points $j \in \mathcal{N}_0$ such that $|\mathcal{K}_j^1| > D$, a similar set of constraints, together with Constraints (3.7), ensure that variables t_j^-, f_j^+ take the correct values:

$$\sum_{k \in U} x_k - D \leq f_j^+ \qquad \forall j \in \mathcal{N}_0 : |\mathcal{K}_j^1| > D, U \subseteq \mathcal{K}_j^1, |U| = d+1 \tag{5.3}$$

$$1 - \sum_{k \in U} x_k \leq t_j^- \qquad \forall j \in \mathcal{N}_0 : |\mathcal{K}_j^1| > D, U \subseteq \mathcal{K}_j^1, |U| = |\mathcal{K}_j^1| - D \tag{5.4}$$

Constraints (5.3)-(5.4) replace (3.5)-(3.6). The logic behind these constraints is the same as the one presented above for Constraints (5.1)-(5.2), the only difference being that a data point classified as 1 is a false positive instead of a true positive. For all data points $j \in \mathcal{N}_0$ such that $|\mathcal{K}_i^1| \leq D$, we can fix variables $t_j^- = 1$.

Once again, if $W_T^- > W_F^+$, Constraints (5.4) are unnecessary, and if $W_T^- < W_F^+$, constraints (5.3) may be dropped. If $W_T^- = W_F^+$ we need both.

Notice that given these constraints variables $t_i^+, f_i^-, t_j^-, f_j^+$ do not need not be binary, only nonnegative. Their integrality is ensured by any feasible solution that does not violate any of Constraints (5.1)-(5.4). We then replace Constraints (3.9)-(3.10) with:

$$t_i^+, f_i^- \geq 0 \qquad \forall i \in \mathcal{N}_1 \tag{5.5}$$

$$t_j^-, f_j^+ \geq 0 \qquad \forall j \in \mathcal{N}_0 \tag{5.6}$$

The number of possible different values for D is finite with $1 \leq D < K$. By solving this formulation for every possible value of D we can take the best solution among all as the global optimum.

In summary, for $1 \leq D < K$ the cutting-planes formulation with a fixed threshold is to optimise (3.1) subject to (3.4), (3.7), (3.8) and (5.1)-(5.6). Depending on how the weights W_T^-, W_F^+, W_T^+ and W_F^- are defined some of constraints (5.1)-(5.4) are unnecessary.

5.1.1 Separation algorithm

We initiate the model without any of Constraints (5.1)-(5.4) since there are an exponential number of them. The separation of all four classes of constraints can be done by inspection in a straightforward procedure. The algorithm works as follows:

1. Initiate the model without any constraints of type (5.1)-(5.4).

2. For every node in the branch-and-bound tree, let $\hat{x}_k, \hat{t}_i^+, \hat{f}_i^-, \hat{t}_j^-, \hat{f}_j^+$ be the variables values in the current linear relaxation.

   a) Constraints (5.1): For $i \in \mathcal{N}_1 : |\mathcal{K}_i^1| > D$

      i. Consider a sorted list of classifiers by descending values of $\hat{x}_k$ (ties broken arbitrarily). Let U be a set containing the first $D+1$ classifiers in this list. If $\sum_{k \in U} \hat{x}_k > \hat{t}_i^+ + D$, add the corresponding constraint to the model.

   b) Constraints (5.2): For $i \in \mathcal{N}_1 : |\mathcal{K}_i^1| > D$

      i. Consider a sorted list of classifiers by ascending values of $\hat{x}_k$ (ties broken arbitrarily). Let U be a set containing the first $|\mathcal{K}_i^1| - D$ classifiers in this list. If $\sum_{k \in U} \hat{x}_k < 1 - \hat{f}_i^-$, add the corresponding constraint to the model.

   c) Constraints (5.3): For $j \in \mathcal{N}_0 : |\mathcal{K}_j^1| > D$

      i. Consider a sorted list of classifiers by descending values of $\hat{x}_k$ (ties broken arbitrarily). Let U be a set containing the first $D+1$ classifiers in this list. If $\sum_{k \in U} \hat{x}_k > \hat{f}_j^+ + D$, add the corresponding constraint to the model.

   d) Constraints (5.4): For $j \in \mathcal{N}_0 : |\mathcal{K}_j^1| > D$

      i. Consider a sorted list of classifiers by ascending values of $\hat{x}_k$ (ties broken arbitrarily). Let U be a set containing the first $|\mathcal{K}_i^1| - D$ classifiers in this list. If $\sum_{k \in U} \hat{x}_k < 1 - \hat{t}_j^-$, add the corresponding constraint to the model.

3. If any violated constraint is added to the model, re-solve the current node. Otherwise, move on to the next node.

Sorting $\hat{x}_k$ can be done once at the beginning of the procedure with cost $O(K \log(K))$. Assuming $N > K$ the separation algorithm is polynomial with complexity $O(NK)$, which includes the cost of cycling through all data points and computing $\sum_{k \in U} \hat{x}_k$.

5.2 Variable threshold reformulation

The first cutting-planes reformulation simplifies the original problem by fixing the classification threshold. We here modify such that the threshold is not fixed, and where we are effectively solving the same problem as the first formulation. For that we need extra binary variables $L_d = 1$ if the threshold is $1 \leq d < K$, 0 otherwise. To ensure that only one threshold value is selected, we need the following constraint:

$$\sum_{d=1}^{K-1} L_d = 1 \tag{5.7}$$

We need to rewrite Constraints (5.1)-(5.4) without assuming a fixed threshold. We can achieve that by making use of the L_d binary variables. For data points $i \in \mathcal{N}_1$ such that $|\mathcal{K}_i^1| > L^d$, Constraints (5.1) and (5.2) can be rewritten as:

$$\sum_{k\in U} x_k - D\sum_{d=1}^{D} L_d - (D+1)\sum_{d=D+1}^{K-1} L_d \leq t_i^+ \qquad \forall D = 1,\ldots,K-1, i\in\mathcal{N}_1, \quad U \subseteq \mathcal{K}_i^1, |U| = D+1 \tag{5.8}$$

$$L_D - \sum_{k\in U} x_k \leq f_i^- \qquad \forall D = 1,\ldots,K-1, i\in\mathcal{N}_1, \quad U \subseteq \mathcal{K}_i^1, |U| = |\mathcal{K}_i^1| - d \tag{5.9}$$

In Constraints (5.8), the expression $D\sum_{d=1}^{D} L_d - (D+1)\sum_{d=D+1}^{K-1} L_d$ replaces the constant value of D in Constraints (5.1). The logic can be illustrated with an example. Consider $K = 10$ and that exactly 6 classifiers in $\mathcal{K}_i^1$ have been chosen to compose the ensemble. If U is composed of these 6 classifiers, we have that $\sum_{k\in U} x_k = 6$ and $\sum_{\mathcal{K}_i^1\setminus U} x_k = 0$. Suppose $D = 5$. In this case, if $L_d = 1$ for some unique $d \leq 5$ (the others being zero), we have $6 - 5 \leq t_i^+$, which would correctly ensure that $t_i^+ = 1$ so that the constraint is not violated.

On the other hand, if $L_d = 1$ for some unique $d > 5$, we would have $6 - 6 \leq t_i^+$. In this case t_i^+ can be zero (but could be one as well). However, since $\sum_{k\in\mathcal{K}_i^1\setminus U} x_k = 0$ in the corresponding Constraint (5.9) we would have the expression $1 - 0 \leq f_i^-$. In this case $f_i^- = 1$, and due to Constraints (3.4) we know that $t_i^+ = 0$.

For every $i \in \mathcal{N}_1$ such that $\mathcal{K}_i^1 = \emptyset$, we fix variables $f_i^- = 1$, and for every i such that $\mathcal{K}_i^1 = \mathcal{K}$ we fix variables $t_i^+ = 1$.

For data points $j \in \mathcal{N}_0$, we use the same logic as above. We ensure correct values for variables t_j^-, f_j^+ with the following constraints:

$$\sum_{k\in U} x_k - D\sum_{d=1}^{D} L_d - (D+1)\sum_{d=D+1}^{K-1} L_d \leq f_j^+ \qquad \forall D = 1,\ldots,K-1, j\in\mathcal{N}_0, \quad U \subseteq \mathcal{K}_j^1, |U| = D+1 \tag{5.10}$$

$$L_D - \sum_{k\in U} x_k \leq t_j^- \qquad \forall D = 1,\ldots,K-1, j\in\mathcal{N}_0, \quad U \subseteq \mathcal{K}_j^1, |U| = |\mathcal{K}_j^1| - D \tag{5.11}$$

Also, for every $j \in \mathcal{N}_0$ such that $\mathcal{K}_i^1 = \emptyset$, we fix variables $t_i^- = 1$ and for every j such that $\mathcal{K}_j^1 = \mathcal{K}$ we fix variables $f_i^+ = 1$.

To summarise the cutting-planes reformulation with a variable threshold is to optimise (3.1) subject to (3.4), (3.7), (3.8) and (5.5)-(5.11).

5.2.1 Separation algorithm

We initiate the model without any of Constraints (5.8), (5.9), (5.10), (5.11) since there are an exponential number of them. The separation of all four classes of constraints can be also done by inspection. The algorithm works as follows:

1. Initiate the model without any constraints of type (5.8)-(5.11).
2. For any given integer solution, take d as the index of the variable L_d whose value in the solution is 1.
3. For every node in the branch-and-bound tree, let $\hat{x}_k, \hat{t}_i^+, \hat{f}_i^-, \hat{t}_j^-, \hat{f}_j^+, \hat{L}_d$ be the variables values in the current linear relaxation.
4. For $1 \leq D < K$:
   a) Constraints (5.8): For $i \in \mathcal{N}_1 : |\mathcal{K}_i^1| > D$
      i. Consider a sorted list of classifiers by descending values of $\hat{x}_k$ (ties broken arbitrarily). Let U be a set containing the first $D + 1$ classifiers in this list. If $\sum_{k \in U} \hat{x}_k > \hat{t}_i^+ + D \sum_{d=1}^{D} \hat{L}_d + (D+1) \sum_{d=D+1}^{K-1} \hat{L}_d$, the corresponding constraint is a candidate for being added to the model.
   b) Constraints (5.9): For $i \in \mathcal{N}_1 : |\mathcal{K}_i^1| > D$
      i. Consider a sorted list of classifiers by ascending values of $\hat{x}_k$ (ties broken arbitrarily). Let U be a set containing the first $|\mathcal{K}_i^1| - D$ classifiers in this list. If $\sum_{k \in U} \hat{x}_k < \hat{L}_D - \hat{f}_i^-$, the corresponding constraint is a candidate for being added to the model.
   c) Constraints (5.10): For $j \in \mathcal{N}_0 : |\mathcal{K}_j^1| > D$
      i. Consider a sorted list of classifiers by descending values of $\hat{x}_k$ (ties broken arbitrarily). Let U be a set containing the first $D+1$ classifiers in this list. If $\sum_{k \in U} \hat{x}_k > \hat{f}_j^+ + D \sum_{d=1}^{D} \hat{L}_d + (D+1) \sum_{d=D+1}^{K-1} \hat{L}_d$, the corresponding constraint is a candidate for being added to the model.
   d) Constraints (5.11): For $j \in \mathcal{N}_0 : |\mathcal{K}_j^1| > D$

i. Consider a sorted list of classifiers by ascending values of $\hat{x}_k$ (ties broken arbitrarily). Let U be a set containing the first $|\mathcal{K}_j^1| - D$ classifiers in this list. If $\sum_{k \in U} \hat{x}_k < \hat{L}_D - \hat{t}_j^-$, the corresponding constraint is a candidate for being added to the model.

5. In the step above, multiple constraints may have been found to be violated. Add one or more of these constraints to the model and re-solve the current node. If no violated constraints are found, move on to the next node.

The procedure has an extra cost of cycling through all possible threshold values, $1 \leq D \leq K-1$. The separation algorithm is polynomial with complexity $O(NK^2)$.

With regards to the implementation, a decision that is not yet clear is what is the best strategy for adding violating constraints. It is possible, for instance, to add all violated constraints found, which can perhaps lower the linear relaxation bound quicker but may make solving the linear relaxation slower. On the other hand, it is possible to add only the most violated constraint of each type, which adds less overhead to solving the linear relaxation but may be slower in reducing the optimality gap. Further experiments are necessary to determine the best option.

5.2.2 Further constraints

A few extra constraints can be added to possibly reinforce the cutting planes formulation with a variable threshold. We present them below.

For $i \in \mathcal{N}_1$, if $|\mathcal{K}_i^1| \leq d$ and $L_d = 1$ we must have that $f_i^- = 1$. In other words, if we use a threshold d that is larger than the number of classifiers that correctly classify i, then i must be a false negative. This can be represented with the following constraints:

$$L_d \leq f_i^- \qquad \forall i \in \mathcal{N}_1, d = |\mathcal{K}_i^1|, \ldots, K-1 \tag{5.12}$$

Equivalently, for $j \in \mathcal{N}_0$, if d is larger than the number of classifiers that wrongly classify j as positive, then j must be a true negative:

$$L_d \leq f_j^+ \qquad \forall j \in \mathcal{N}_0, d = |\mathcal{K}_j^1|, \ldots, K-1 \tag{5.13}$$

Both these sets of constraints may help in reducing the number of Constraints (5.8)-(5.11) that need to be added to the model.

Suppose also that the number of selected classifiers is $K' < K$. If $L_d = 1$ for any $K' \leq d \leq K-1$, then all data points are classified as negative. All these possible feasible solutions are identical, with the same objective value. We can discard all but one of them to reduce the solution space with the following constraints:

$$L_e \leq L_d + \sum_{k=1}^{K} x_k - d + d \sum_{c=1}^{d-1} L_c \qquad d = 1, \dots, K-2, e = d+1, \dots, K-1 \tag{5.14}$$

The constraints above work as follows. If $\sum_{k=1}^{K} x_k > d$, then Constraints (5.14) are always satisfied. If $\sum_{k=1}^{K} x_k = d$, then $L_e \leq L_d$ for any $e > d$, effectively eliminating multiple identical solutions as $\sum_{d=1}^{K-1} L_d = 1$. If $\sum_{k=1}^{K} x_k < d$, then there must be $L_c = 1$ for $c < d$ as the corresponding Constraints (5.14) for $c = d$ ensure that $L_d = 0$ for any $d > c$. In this case the term $d \sum_{c=1}^{d-1} L_c$ ensures that the right-hand side is not negative and the constraint is not violated.

5.3 Summary

We have presented two cutting-planes reformulations. The first is to optimise (3.1) subject to (3.4), (3.7), (3.8) and (5.1)-(5.6). This reformulation assumes a fixed threshold D. Since the possible values for the threshold are finite, this formulation must be solved for each value $1 \leq D < K$. If all subproblems are solved than the optimal solution is the best among them.

The second formulation is a modification of the first in which the threshold is once again variable. Its goal is to optimise (3.1) subject to (3.4), (3.7), (3.8) and (5.5)-(5.11). We will also test whether the inclusion of (5.12)-(5.14) helps in speeding up the solution process.

5.4 Considerations

While we do not present computational results for the reformulations in this thesis, we have some considerations regarding their implementation and some steps we are currently undertaking.

One of the most important aspects in implementing these formulations is to decide on the policy of adding violated constraints to the model. In the second reformulation, we have the potential to find NK violated constraints in a single execution of the separation algorithm. Generally adding many cuts in multiple iterations can considerably slow down the linear relaxation solution procedure. Moreover, there is a possibility of several cuts being added which will not be tight in the optimal solution.

For instance, if in the optimal solution $L_d = 1$ for some d, then all Constraints (5.8)-(5.11) with L_e, $e \neq d$, are unnecessary. Thus finding the best balance in adding the most promising violated constraints is a challenge in this research.

Also, giving higher priority in branching to the L_d binary variables may help the solver into focusing on the most promising candidates for threshold earlier on, while helping reduce the number of unnecessary violated constraints added to the model. Whether that will hold remains an open question.

Another important aspect is to prove mathematically whether the second reformulation presented here is "stronger" than the first. By stronger we mean that after solving the first linear relaxation without any fixed variables, we consistently find stronger upper bounds. We will attempt, in our work, to prove this mathematically in case it is true, but we have not yet done so.

Initially however we can get an idea of whether the reformulation is stronger or not by running a set of experiments in which we solve both linear relaxations and compare to see which, if any, is consistently stronger.

We mention mostly the second reformulation since, as previously stated, we need to solve the first reformulation multiple times to obtain the global optimum. In this case it is not straightforward how we should compare it to the original formulation. If the global optimum is achieved than we can compare required computational times, but other important metrics such as global optimality gap or the bound provided by the linear relaxation cannot be easily calculated. We thus intend to dedicate more effort to the second reformulation, which is directly comparable. Nevertheless we opted to keep both in this text as the first reformulation acts as a "bridge" to the more mathematically complex second reformulation.

5.5 A new diversity model based in lazy constraints

We introduced in Section 3.2 a set of constraints in order to control diversity in our initial model. We considered the Pairwise Failure Crediting (PFC) as our diversity measure. Pairwise measures are those that are calculated based on a relation of a pair of classifiers. The PFC is calculated between the pair of the individual classifiers, and as we calculated it statically before the optimisation and used its values as a parameter, we were able to create a linear integer model that requires $\binom{K}{2}$ extra variables and a similar number of extra constraints that as showed in Section 4.2 causes a negligible decrease in solution quality.

There are alternative ways to calculate diversity. For instance, instead of computing a measure between each pair of classifiers, we can measure the relation between the output of an individual classifier and the output of the current ensemble. Examples of such diversity measures are complementariness [33], concurrency [3] and Uncertainty Weighted Accuracy (UWA) [39].

Such diversity measures that uses the current ensemble output cannot be computed statically as we do with the PFC measure. In this section we use a generic lazy constraints approach which allows controlling such diversity measures. Here we describe our idea using the formulation presented in Chapter 3. In our model, the output of the classifiers are in the A and B matrices, of $N_0 \times K$ and $N_1 \times K$ dimensions. The ensemble output is accessed by the variables t_i^+, f_i^-, vectors of size N_1, and t_j^-, f_j^+ of size N_0. For this reason, to create any diverse constraints that considers the entire ensemble, we would need to use the values of the integer variables t_i^+, f_i^-, t_j^-, f_j^+ in relation with K classifiers. So we would need at least $2K(N_0N_1)$ new constraints, a number that should be increased depending on the diverse measure. Depending on the size of the dataset, adding those number of constraints that are dependent of integer variables may really complicate the model and decrease its solution quality.

For this reason, we think that the use of lazy constraints may contribute to avoid this problem. In our idea, when a given integer solution is found, we verify if this solution is undesirable from the point of view of diversity. If it is, we can take some decision based on it. For instance, we can use a constraint that will not allow this specific ensemble U to be chosen as our final ensemble. We can do it using the Constraint 5.15.

$$\sum_{k \in U} x_k + \sum_{k \in \mathcal{K} \setminus U} (1 - x_k) \leq K - 1 \tag{5.15}$$

Observe that the Constraint (5.15) will only prohibit the current ensemble U to be a possible solution. The inclusion or exclusion of any classifier is allowed.

The method introduced above is interesting because it is independent of diversity measure. As the computation of the diversity measure is done via lazy-constraints, we do not need to create constraints to compute this measure. This way we can easily adapt it to any suitable measure we find. We intend to implement this methodology as a future work and test it with alternative diversity metrics.

Chapter 6

Conclusions and future directions

In this thesis we proposed an IP approach for the problem of selecting a subset of classifiers in ensemble learning. The objective is to maximise a weighted function of the patterns in the confusion matrix. In order to combine performance and diversity criteria, we also proposed linear constraints to enforce minimum diversity levels. We observed that state-of-the-art solvers can find good solutions in reasonable computational times for the chosen publicly available datasets. The IP approach is, in our view, able to provide a flexible exact algorithm (with regards to both the choice of performance metric and desired diversity levels) which can also be used as a heuristic by enforcing a time limit to the solver. This approach has also the additional advantage of providing bounds on optimal values.

We compared our formulation to three benchmarks: the full ensemble and two of the most popular algorithms found in literature. We used a stratified 10-fold cross validation procedure and evaluated the effect of enforcing minimum diversity levels, varying the weights assignments of the objective function and using majority voting as our classification threshold. We observed slightly lower variability and improved overall performance compared to the benchmarks algorithms when the formulation without minimum diversity levels is applied. We were able to reduce that variability while also improving performance by including diversity constraints in the formulation. In the case of adding majority voting constraints, we improved our results both with and without diversity constraints, but increasing variability.

We also tested the use of weights in our formulation, tackling the class imbalance problem. We were able to improve the AUC results with the θ-weighted formulation in comparison with our formulation without weights, also beating the benchmarks. Using a balanced accuracy as our metric, the benefits of the weights were even more highlighted.

When it comes to the problem of overfitting, we verified that our formulation really has a higher deterioration in the out-of-sample results compared to the in-sample ones. This is expected as our formulation finds optimal in-sample solutions while most algorithms in literature are suboptimal in nature. However, even with this problem, our formulation was still able the get better overall performance.

In our view, those results suggest that our approach is competitive and its flexibility can be beneficial when adapting to datasets with different characteristics.

As future work we intend to experiment with different criteria and larger datasets, as well as implementing other methods in literature for a more thorough out-of-sample performance comparison. We also intend to explore our cutting planes reformulation in order to improve general computational times, as outlined in Section 5.4. While for the instances tested we considered that the computational performance was satisfactory, we believe it will deteriorate when we include larger datasets. We intend also to research other IP techniques and formulation-based heuristics for both finding good solutions quickly and solving the formulation to optimality faster. Other lines of research include the study of alternative diversity based models (e.g. the model outlined in Section 5.5) and presenting an alternative formulation for the problem of maximising AUC.

Bibliography

[1] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2006.

[2] Bart Bakker and Tom Heskes. Clustering ensembles of neural network models. *Neural Networks*, 16(2):261–269, 2003.

[3] Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer, and W.Philip Kegelmeyer. Ensemble diversity measures and their application to thinning. *Information Fusion*, 6(1):49–62, 3 2005.

[4] Saman Bashbaghi, Eric Granger, Robert Sabourin, and Guillaume Alexandre Bilodeau. Robust watch-list screening using dynamic ensembles of SVMs based on multiple face representations. *Machine Vision and Applications*, 28(1-2):219–241, 2017.

[5] Urvesh Bhowan, Mark Johnston, Mengjie Zhang, and Xin Yao. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation*, 17(3):368–386, 2012.

[6] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[8] A. S. Britto, R. Sabourin, and L. E.S. Oliveira. Dynamic selection of classifiers - a comprehensive review. *Pattern Recognition*, 47(11):3665–3680, 2014.

[9] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18. ACM, 2004.

[10] A. Chandra and X. Yao. Ensemble learning using multi-objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms*, 5(1):417–445, 2006.

[11] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[12] J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[13] CPLEX Optimizer. IBM. Available from https://www.cplex.com, last accessed November 15th, 2019.

[14] R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41(May):195–216, 2018.

[15] Q. Dai and X. Han. An efficient ordering-based ensemble pruning algorithm via dynamic programming. *Applied Intelligence*, 44(4):816–830, 2016.

[16] J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine learning*, pages 233–240, 2006.

[17] R. P. W. Duin. The combining classifier: to train or not to train? *Object recognition supported by user interaction for service robots*, 2(c):765–770, 2002.

[18] Wei Fan, Fang Chu, Haixun Wang, and Philip S. Yu. Pruning and dynamic scheduling of cost-sensitive ensembles. In *AAAI/IAAI*, pages 146–151, 2002.

[19] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[20] Everlandio RQ Fernandes, André CPLF de Carvalho, and André LV Coelho. An evolutionary sampling approach for classification with imbalanced data. In *IJCNN'15. International Joint Conference on Neural Networks.*, pages 1–7. IEEE, 2015.

[21] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[22] J. H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American statistical Association*, 76(376):817–823, 1981.

[23] G. Giacinto, F. Roli, and G. Fumera. Design of effective multiple classifier systems by clustering of classifiers. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 160–163. IEEE, 2000.

[24] Tin Kam Ho. The Random Subspace Method for Constructing Decision Forest. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[25] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[26] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239, 1998.

[27] Aleksandar Lazarevic and Zoran Obradovic. Effective pruning of neural network classifier ensembles. In *IJCNN'01. International Joint Conference on Neural Networks.*, volume 2, pages 796–801. IEEE, 2001.

[28] M. Lichman. UCI Machine Learning Repository. Available from http://archive.ics.uci.edu/ml, last accessed November 15, 2019, 2019.

[29] Z. Lu, X. Wu, X. Zhu, and J. Bongard. Ensemble pruning via individual contribution ordering. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 871–880. ACM, 2010.

[30] D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *Proceedings of the 14th International Conference of Machine Learning*, volume 97, pages 211–218. Citeseer, 1997.

[31] G. Martínez-Muñoz and A. Suárez. Pruning in ordered bagging ensembles. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 609–616. ACM, 2006.

[32] G. Martínez-Muñoz and A. Suárez. Using boosting to prune bagging ensembles. *Pattern Recognition Letters*, 28(1):156–165, 2007.

[33] Gonzalo Martınez-Muñoz and Alberto Suárez. Aggregation ordering in bagging. In *Proc. of the IASTED International Conference on Artificial Intelligence and Applications*, pages 258–263. Citeseer, 2004.

[34] M. Padberg and G. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch-and-cut. *Operations Research Letters*, 6(1):1–7, 1987.

[35] M. W. Padberg and L. A. Wolsey. Trees and cuts. *North-Holland Mathematics Studies*, 75(1):511–517, 1983.

[36] Ioannis Partalas, Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Ensemble pruning using reinforcement learning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Berlin, Germany, 2006. Springer.

[37] Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. Focused ensemble selection: A diversity-based method for greedy ensemble selection. *Frontiers in Artificial Intelligence and Applications*, 178:117–121, 2008.

[38] Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. Pruning an ensemble of classifiers via reinforcement learning. *Neurocomputing*, 2009.

[39] Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. An ensemble uncertainty aware measure for directed hill climbing ensemble pruning. *Machine Learning*, 81(3):257–282, 2010.

[40] Fu Qiang, Hu Shang-Xu, and Zhao Sheng-Ying. Clustering-based selective neural network ensemble. *Journal of Zhejiang University-Science A*, 6(5):387–392, 2005.

[41] Marina Skurichina and Robert P W Duin. Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis and Applications*, 5(2):121–135, 2002.

[42] E Ke Tang, Ponnuthurai N Suganthan, and Xin Yao. An analysis of diversity measures. *Machine learning*, 65(1):247–271, 2006.

[43] G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective voting of heterogeneous classifiers. In *European Conference on Machine Learning*, pages 465–476. Springer, 2004.

[44] G. Tsoumakas, I. Partalas, and I. Vlahavas. An ensemble pruning primer. In *Applications of supervised and unsupervised ensemble methods*, pages 1–13. Springer, 2009.

[45] Grigorios Tsoumakas, Lefteris Angelis, and Ioann Vlahavas. Selective fusion of heterogeneous classifiers. *Intelligent Data Analysis*, 2005.

[46] L. Xu, B. Li, and E. Chen. Ensemble pruning via constrained eigen-optimization. In *2012 IEEE 12th International Conference on Data Mining*, pages 715–724. IEEE, 2012.

[47] Y. Zhang, S. Burer, and W. N. Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7(Jul):1315–1338, 2006.

[48] Zhi-Hua Zhou and Wei Tang. Selective ensemble of decision trees. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pages 476–483. Springer, 2003.

[49] X. Zhu, X. Wu, and Y. Yang. Dynamic classifier selection for effective mining from noisy data streams. In *4th IEEE International Conference on Data Mining*, pages 305–312. IEEE, 2004.

Appendix A

Description of the classifiers

As mentioned in Chapter 4, we prepared 10 different heterogeneous classifier models, but used up to $K = 100$ different classifiers for creating ensembles. Each model of the 10 models was initiated 10 times with different random seeds and parameters. It is worth reminding that, as mentioned in the paper, we used subsets of the full set of classifiers, varying their sizes in $K = \{10, 20, 30, 40, 60, 80, 100\}$. We set K as multiples of 10 in order to have an equal number of instantiations of each classifier.

For each classifier model, their initialisation differs in two ways: parameter settings and random seeds required by the algorithms. The 10 classifier models are shown below. We include the following information: Model Package, Package Class (`sklearn` package models), GitHub, documentation, version, parameters varied and normalisation (when applicable).

1. XGBoost
    - Package: xgboost
    - GitHub: `https://github.com/dmlc/xgboost`
    - Documentation: `https://xgboost.readthedocs.io/en/latest/`
    - Used version: 0.80
    - Used parameters: objective, booster, num_boost_round, eta, max_depth, subsample, colsample_bytree, colsample_bylevel, colsample_bynode, gamma, min_child_weight
2. LightGBM
    - Package: lightgbm
    - GitHub: `https://github.com/microsoft/LightGBM`
    - Documentation: `https://lightgbm.readthedocs.io/en/latest/`
    - Used version: 2.2.0
    - Used parameters: objective, boosting_type, num_boost_round, learning_rate, max_depth, num_leaves, feature_fraction, bagging_fraction, bagging_freq, min_data_in_leaf, max_cat_to_onehot, lambda_l1, lambda_l2

3. Catboost
    - Package: catboost
    - GitHub: `https://github.com/catboost/catboost`
    - Documentation: `https://catboost.ai/docs/`
    - Used version: 0.11.1
    - Used parameters: iterations, cat_features, learning_rate, l2_leaf_reg, one_hot_max_siz bootstrap_type, Bayesian, subsample, depth, rsm
4. Adaboost
    - Package: scikit-learn
    - Class: sklearn.ensemble.AdaBoostClassifier
    - GitHub: `https://github.com/scikit-learn/scikit-learn`
    - Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html`
    - Used version: 0.19.2
    - Used parameters: base_estimator, n_estimators, learning_rate
5. Gradient Boosting
    - Package: scikit-learn
    - Class: sklearn.ensemble.GradientBoostingClassifier
    - GitHub: `https://github.com/scikit-learn/scikit-learn`
    - Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html`
    - Used version: 0.19.2
    - Used parameters: n_estimators, learning_rate, max_depth, criterion, subsample, max_features, min_samples_split, min_samples_leaf
6. Bagging
    - Package: scikit-learn
    - Class: sklearn.ensemble.BaggingClassifier
    - GitHub: `https://github.com/scikit-learn/scikit-learn`
    - Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html`

- Used version: 0.19.2
- Used parameters: base_estimator, n_estimators, learning_rate, max_features, max_samples

7. Random Forest
    - Package: scikit-learn
    - Class: sklearn.ensemble.RandomForestClassifier
    - GitHub: `https://github.com/scikit-learn/scikit-learn`
    - Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`
    - Used version: 0.19.2
    - Used parameters: n_estimators, criterion, max_features, max_depth, min_samples_spli min_samples_leaf

8. Extra Trees
    - Package: scikit-learn
    - Class: sklearn.ensemble.ExtraTreesClassifier
    - GitHub: `https://github.com/scikit-learn/scikit-learn`
    - Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html`
    - Used version: 0.19.2
    - Used parameters: n_estimators, criterion, max_features, max_depth, min_samples_spli min_samples_leaf

9. Logistic Regression
    - Package: scikit-learn
    - Class: sklearn.linear_model.LogisticRegression
    - GitHub: `https://github.com/scikit-learn/scikit-learn`
    - Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`
    - Used version: 0.19.2
    - Used parameters: max_iter, solver, penalty, dual, fit_intercept, C, tol

10. Multilayer Perceptron

- Package: scikit-learn
- Class: sklearn.neural_network.MLPClassifier.html
- GitHub: `https://github.com/scikit-learn/scikit-learn`
- Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`
- Used version: 0.19.2
- Used parameters: max_iter, hidden_layer_sizes, activation, solver, learning_rate, learning_rate_init, momentum
- Normalisation: sklearn.preprocessing.StandardScaler with default parameters - `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

We used a stratified 10-Fold procedure to evaluate the proposed method, described in Section 4.3. To generate different folds we varied the random seeds required shuffle the datasets.

Random number generation is executed twice (classifier initialisation and dataset shuffling). For the shuffling we used random seeds $10, 20, 30, \ldots, 100$. The seeds used in the classifiers depended on the stratified 10-fold seed. For instance, if $K = 100$ and we used seed 10, then the seeds for each of the 10 classifiers of each model were 10, 11, $\ldots$ 19.