Chapter 20
A Multi-objective Meta-Analytic Method for Customer Churn Prediction

Mohammad Nazmul Haque, Natalie Jane de Vries, and Pablo Moscato

Abstract The term 'metaheuristic' was introduced in 1986 as a way to label 'a higher-level procedure designed to guide a lower-level heuristic or algorithm' to find solutions for tasks posed as mathematical optimization problems. Analogously, the term 'meta-analytics' can be used to refer to a higher-level procedure that guides ad hoc data analysis techniques. Heuristics that guide ensemble learning of heterogeneous classifier systems would be one of those procedures that can be referred to as 'meta-analytics'. In general, researchers use single-objective approaches for ensemble learning. In this contribution we investigate the use of a multi-objective evolutionary algorithm and we apply it to the problem of customer churn prediction. We compare the results with those of a symbolic regression-based approach. Each has its own merits. While the multi-objective approach excels at prediction, it lacks in interpretability for business insights. Oppositely, the symbolic regression-based approach has lower accuracy but can give business analysts some actionable tools. Depending on the nature of the business scenario, we recommend that both be employed together to maximize our understanding of consumer behaviour. High-quality individualized prediction based on multi-objective optimization can help a company to direct a message to a particular individual, while the results of a global symbolic regression-based approach may help large marketing campaigns or big changes in policies, cost structures and/or product offerings.

Keywords Churn · Customer churn prediction · Ensemble of classifiers · Ensemble learning · Genetic programming · Multi-objective ensemble · NSGA-II algorithm · Symbolic regression

M. N. Haque (✉) · N. J. de Vries · P. Moscato
School of Electrical Engineering and Computing, The University of Newcastle, Callaghan, NSW, Australia
e-mail: Mohammad.Haque@newcastle.edu.au; natalie.devries@newcastle.edu.au; Pablo.Moscato@newcastle.edu.au

P. Moscato, N. J. de Vries (eds.), *Business and Consumer Analytics: New Ideas*,
https://doi.org/10.1007/978-3-030-06222-4_20

20.1 Introduction and Motivation

Since the early 1980s, machine learning research has been increasingly adopted to solve many problems in business and consumer analytics. The pace of the field is relentless with new breakthroughs being published almost on a daily basis. Needless to say, there is a wealth of information available on the Internet. Since the establishment of the World Wide Web in the early 1990s, knowledge boundaries have dissolved and people can now have access to software and source code from almost anywhere in the world just by browsing public domain repositories. The adoption of languages like Java, R, Python, etc. are also contributing to the development of large collections of machine learning and classification software which is 'top-of-the-shelf' and ready to use for data analytic objectives. The question for this particular field is then which classification algorithms would be most useful for a given problem when we have such an abundance of alternatives.

The path to answering this question is perhaps obvious. We need to conduct research in the area of meta-analytics, that is, methodologies which aim at guiding individual analytic methods to improve the final performance by exploiting synergies between the individual classifiers. A prime candidate for such research is the area of Ensemble Learning. Broadly, ensemble learning is the process by which multiple models, such as classifiers (as in our case) or experts, are strategically generated and combined to solve a particular computational intelligence problem.

We should be aware that the primary objective in data classification is to finally deliver a method that consistently predicts the right target when confronted with samples coming from a particular distribution. The system has been trained with an independent set of samples and thus good performance relative to these new samples is seen as a good indicator of its generalization capability. During the past decade, researchers have been increasingly accepting the notion that ensembles of classifiers can be a useful way to improve the generalization capability in problems related to prediction and regression.

In spite of the fact that theoretical results indicate that ensembles could outperform the generalization performance of their individual members (called 'base classifiers'), selecting such a set—a group that 'works together' to meet the desired theoretical criteria (combining all accurate and diverse base classifiers) given a particular classification problem—is not an easy task [11]. It may be, in a certain sense, what mathematicians describe as an 'ill posed' problem. However, it is clear that a meta-analytic procedure embodies the appropriate elements for selecting such a group. This provides a strong direction for future research to assure a consistent flow of new innovative designs for base classifiers. Beyond the first objective of classification/prediction, we are confronted with the objective of selecting a suitable set of classifiers. Analogous to the problem of over-fitting in machine learning, including too many base classifiers may achieve high-performance in a training set but yield poor generalization ability. This means that we would need to identify, for a given analytic request, a set of base classifiers that generalizes well and does not have too many members. This introduces another dimension that we will make clear in Sects. 20.6.3 and 20.8 that ends this chapter. For the purpose of interpretability

in business and consumer insight it is also desirable that the classifiers are of a relatively small number. Each of the classifiers can be interpreted as a 'model' of our business and the variables in it can allow some action which will result in better decision-making and outcomes. Having too many models may obscure the knowledge that can be derived from the whole process.

From the discussion above, and from a meta-analytics perspective, we are dealing with a multi-objective scenario: that of trying to maximize the generalization capability of an ensemble of classifiers while at the same time reducing the number of members in its set. We are thus recognizing an emerging trend. There have been advances in single-objective approaches (mostly accuracy-based), but now some researchers are turning to multi-objective optimization for further improvements by generalizing the ensemble of classifiers [4, 11, 16, 18]. Some accuracy-based approaches may have some problems with classification problems which have classes that are very imbalanced, as, for instance, when we have to discriminate between two classes and one class only has 10% of the samples represented in the training set. This means that other types of objective guidance functions should be selected for multi-objective optimization (MOO). Our contribution presents a case study of this type of problem domain in which we show how a meta-analytic procedure can handle multiple conflicting criteria for the design of an ensemble of classifiers in a challenging business analytics problem.

20.2 Churn Prediction: An Important Issue for Customer Relationship Management

It is well-known among business and marketing managers that obtaining new customers is a more difficult feat than maintaining existing ones. It is also more costly and time-consuming for the business. Although customer relationship management (CRM) is not always easy either, hanging on to existing customers is usually more feasible than trying to attract new customers (who may not know about your business or product). This makes customer churn an interesting research area. A customer is a 'churner' when he/she leaves the company as a customer. This concept is more applicable to service companies when long-term and ongoing relationships between the company and the consumer are more common, as in the case of banks or phone or internet providers. Naturally, businesses would like to be able to predict these churners or, at the very least, differentiate them from the customers who remain (non-churners) so that certain insights can be made into why these customers are churning. This scenario provides a natural setting for applying multi-objective optimization. Specifically, we investigate the problem of forecasting churners and non-churners of a telecom services company, drawing on the Churn telecom service customer dataset, which is publicly available online.[1] Details of the dataset can be found in the dataset description in Chap. 26.

[1] churn.txt inside the compressed file at: http://dataminingconsultant.com/DKD2e_data_sets.zip.

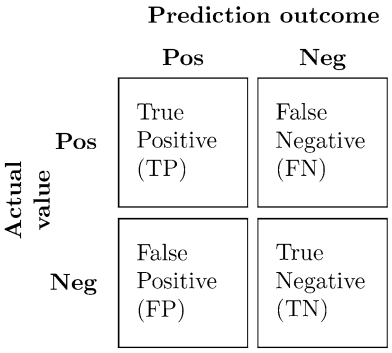

Fig. 20.1 A confusion matrix summarizes all possible outcomes in a binary-classification problem (TP, FP, FN and TN). In our case a TP means a churner predicted correctly to churn, a FP is a remaining customer that was predicted as a churner, a FN is a churner that was not predicted to churn and a TN is a remaining customer that was correctly predicted not to churn

Our objective is to predict 'churners' (those customers who left the company) and differentiate them from the 'non-churners' (those who remained loyal customers to the company). This dataset is a binary-class dataset containing 3333 samples and 20 features [24]. We split the dataset into training and testing sets using threefold cross validation (CV) of the original dataset (use 1 as the seed for randomly splitting with WEKA [13]). We used stratified-folds to keep the class distribution similar to the original dataset while separating them into training and testing folds. After doing the threefold CV, we get three pairs of training and testing fold datasets. The number of samples in each Train Folds is 2222 (1900, 322) and Test Folds is 1111 (950,161).

20.3 Classification Performance Measures

The importance of a classifier's prediction performance is critical. The prediction performance of a classifier is usually reported using a two row by two column matrix, widely known as a *confusion matrix* [29]. The four outcomes of a classification problem can be summarized in the confusion matrix as shown in Fig. 20.1.

Here, the outcomes are labelled as belonging to either the positive (Pos) or the negative (Neg) class. If the outcome from a prediction and the actual value is Pos, then it is called a true positive (TP); however, if the actual value is Neg, then it is said to be a false positive (FP). Conversely, when both predicted and actual outcomes are Neg, then it is denoted as a true negative (TN). It is denoted as a false negative (FN) when the prediction outcome is Neg, while the actual value is Pos.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{20.1}$$

$$Recall/Sensitivity = \frac{TP}{(TP + FN)} \tag{20.2}$$

$$Specificity = \frac{TN}{(TN + FP)} \tag{20.3}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{20.4}$$

$$F\text{-}Measure = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{20.5}$$

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)\,(TP + FN)\,(TN + FP)\,(TN + FN)}} \tag{20.6}$$

Different measures of classifier performance can be calculated from the confusion matrix, as shown in Eqs. (20.1)–(20.6). The classification accuracy ($Accu$) is one of the most significant metrics. Although classification accuracy is a measure used for comparing the prediction performance, it is not suitable for use with imbalanced data when the sample sizes of the two classes differ greatly. It only considers the proportion of correct classifications without considering the class distribution. This can result in the poor rating for the minority class being easily overwhelmed by the correct classification of the majority class. We note that identifying the minority classes can actually be an important outcome in many cases, such as fraud detection, cancer detection and intrusion detection. Thus, the more appropriate evaluation criteria are those that use all values from the confusion matrix for binary-classification problems. Other popular measures for comparing performance of different classifiers are based on *sensitivity* and *specificity*. Sensitivity (SEN) is the probability of predicting a positive outcome when the true state is positive, whereas specificity ($SPEC$) is the probability of predicting a negative outcome when the true state of a case is negative. While there is no perfect way of describing the confusion matrix of TP, FP, FN and TN by a single number, the Matthews Correlation Coefficient (MCC in Eq. (20.6)) quantifies the strength of the classifications using the confusion matrix. The MCC takes into account both the sensitivity and specificity, and can often provide a more balanced accuracy assessment of the model [10]. An MCC with a higher value indicates better predictions. It is commonly used as a performance measure for imbalanced datasets [15] in different fields of machine learning, including bioinformatics [14]. Notably, the MCC was chosen as a measure classifier performance in the initiative FDA MAQC-II led by the USA, which was based on microarray gene expression and genotyping data [27]. Further, for the analysis of the confusion matrix, the experimental results obtained by Jurman et al. [17] indicate that MCC is an ideal measure for practical classification. Thus, we have selected MCC as our measure of classification performance.

20.4 A Brief Introduction to Multi-objective Optimization in Ensemble Learning

The core principle of ensemble learning is to weigh several individual pattern classifiers, and then combine them to gain better accuracy than can be obtained by using each of them separately. In the machine learning paradigm, ensemble data mining methods strategically advance the power of committee methods, or combine models to achieve better prediction accuracy than any of the individual models could achieve. The basic goal when designing an ensemble is to use independent models so that the combined model will produce a better performance. Ensemble methods have been used by researchers from various disciplines such as pattern recognition, statistics and machine learning.

In addition to better prediction, in real-world scenarios we are often required to give solutions that optimize more than one objective function. Analogous to practising the art of 'feature engineering', applied mathematicians and computer scientist have to select objective functions that may best address the actual task they are required to perform. For some problems these functions are more natural, but in other cases the practitioners may need to 'engineer' which function or combination of those best suits the problem domain.

In some cases, the more useful objective functions are quite explicit but also rather conflicting. For example, we may be required to maximize a certain measure of performance while at the same time minimizing the cost to achieve this performance. At the same time we may need to maximize reliability while minimizing risk and execution time. In those cases, we typically cannot find a single final solution that is optimal for all objectives.

20.4.1 Multi-objective Optimization in a Nutshell

A multi-objective optimization problem (MOP) can be defined as the problem of finding optimal feasible solutions that optimize more than one conflicting objectives:

$$\textbf{(Minimize)} \left[f_1 (\mathbf{x}) , f_2 (\mathbf{x}) , \cdots , f_p (\mathbf{x}) \right] \quad \mathbf{x} \in \mathbf{S}, \tag{20.7}$$

where f_i, with $i = 1, 2, \cdots , p$ is the i-th objective function and $\mathbf{x}$ is a solution from the set of feasible solutions $\mathbf{S}$. In multi-objective optimization we are interested in more than one feasible solution for the optimization problem. The feasible solutions we seek are called *Pareto-optimal* solutions.

A solution $\mathbf{x}^* \in \mathbf{S}$ is called *Pareto-optimal* if there does not exist another solution that dominates it by virtue of having a smaller or equal objective value. A solution

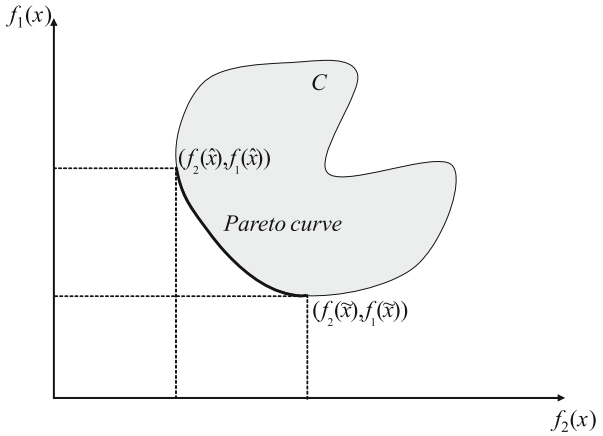

Fig. 20.2 An example of Pareto-front. The figure is adapted from [3] with permission

is called *non-dominated* if all other solutions $\mathbf{x} \in \mathbf{S}$ have higher value for at least one objective function f_i or have the same value for all the objective functions [3].

For a two-objective problem we can plot the objective values of solutions (also denoted as *Pareto-front*) in a x-y plane. An example of a *Pareto curve* is shown in Fig. 20.2, where all the points on the Pareto curve between $\left(f_2\left(\hat{x}\right), f_1\left(\hat{x}\right)\right)$ and $(f_2(\tilde{x}), f1(\tilde{x}))$ are called *non-dominated* points.

20.5 Our Multi-objective Meta-Analytic Method for Churn Prediction

In this section we specify the different components of our approach: the different objective functions to be used, the type of classifiers to be employed and so forth.

We have chosen 28 heterogeneous base classifiers available in the WEKA data mining software, version 3.7 [13]. We only chose those single classifiers which are not meta-classifiers and are capable of handling the churn dataset. The list of classifiers is given in Table 20.1.

20.5.1 Objective Selection for Multi-objective Optimization

Most of the research on ensemble generation using multiple-objectives is confined to a single scalar function using a hyper-parameter [16]. It is a common practice to tune the hyper-parameter (λ) to balance between two commonly used objectives, Obj_A and Obj_B to get the scalar value of objectives Obj_{Sc} as:

$$Obj_{Sc} = Obj_A + \lambda Obj_B. \tag{20.8}$$

The conversion of multi-objective into single objective usually introduces some weakness. A predefined hyper-parameter is required to be set in order to achieve the right balance between Obj_A and Obj_B. Moreover, transforming this outcome into single-objective approach is limited to type of the problem. We do need the multi-objective problem to actually optimize relative to multiple conflicting objectives. Best individuals from a multi-objective evolution will form the Pareto front of a size equal to the number of objectives.

Table 20.1 List of 28 base classifiers to be used for the experiments

Type	Count	Base classifiers
Bayes	2	`BayesNet`
		`NaiveBayes`
Support vector machine	2	`SMO`
		`SPegasos`
Linear	3	`Logistic`
		`SimpleLogistic`
		`SGD`
Neural network	3	`MLPClassifier`
		`RBFNetwork`
		`VotedPerceptron`
Decision tree	7	`ADTree`
		`BFTree`
		`HoeffdingTree`
		`J48`
		`LADTree`
		`REPTree`
		`PART`
Decision stump	1	`DecisionStump`
Trees	4	`FT`
		`LMT`
		`RandomTree`
		`SimpleCart`
Nearest neighbours	1	`IBk`
Rule learner	3	`ConjunctiveRule`
		`JRip`
		`Ridor`
Decision table	1	`DecisionTable`
Feature interval	1	`VFI`

20.5.1.1 Definition of Objective Functions

In our case, we analyse three commonly used objective values in the literature, namely classification performance measure, diversity score and ensemble size. Objectives based on these values conflict with each other. We define the objective functions for our Multi-Objective Ensemble of Classifier (MO-EoC) as follows:

Classification Performance Measure: Utilizing the classification performance measure, we maximize the average Matthews Correlation Coefficient (MCC) score for the ensemble of classifier as described in [15]. Hence, the objective function can be written as:

Objective 1 ***MCC:*** *Given an ensemble* $\mathbb{E}$ *with k base classifiers. For each base classifier* $\mathbb{C}_i$ *in the ensemble*

$$\textbf{(Maximize)} \quad Obj_{mcc}(\mathbb{E}) : \frac{1}{k}\sum_{i=1}^{k} MCC\left(\mathbb{C}_i\left(\mathbb{T}_s\right) \in \mathbb{E}\right) \qquad (20.9)$$

$$subject\ to \quad MCC\left(\mathbb{E}\right) \geq 0.0,$$

where the MCC score is calculated using 60–40 split on the selected feature subset $\mathbb{T}_s$*. A valid solution should not perform worse than a random predictor (with an MCC score equals to at least 0.0).*

The outcome of optimizing this objective will find the ensemble combination providing a maximum average MCC score. Hence, the optimization of Obj_{mcc} function will be a maximization problem.

Diversity Measure of Base Classifiers: In the classification problem, the diversity score represents the agreement or disagreement among the base classifiers' decisions. The most diverse classifiers will predict different labels for the same input data. In the literature about the ensemble of classifiers, the diversity among base classifiers is not well defined as classification performance [31]. Furthermore, [19] experimented with available diversity measures and found that some of them may mislead the classification learning. This work divides the diversity measures into two categories: pairwise and non-pairwise diversity.

In the case of pairwise diversity measures, the score is calculated for the pair of base classifiers. The calculation of pairwise diversity measure becomes computationally expensive for ensembles created with a large number of base classifiers and also for large datasets. Q-statistics, correlation coefficient and k-statistics are commonly used pairwise diversity scores [1, 26]. On the other hand, the non-pairwise diversity score is calculated over a set of classifiers for training performance. The non-pairwise diversity measures can be calculated easily as a group for a large number of base classifiers in contrast to the pairwise measures which consider each pair of base classifiers. Kohavi–Wolpert variance,

inter-rater agreement, generalized diversity and entropy are commonly used non-pairwise measures of diversity [5, 25]. In our case, we have 28 heterogeneous base classifiers, which is the largest ensemble of classifiers considered so far.

We use the widely used entropy score for ensemble of classifiers as our measure of diversity [19]. The value of diversity for an ensemble $\mathbb{E}$ on training dataset $\mathbb{T}_s$ is calculated as:

$$\frac{1}{m}\sum_{j=1}^{m}\frac{1}{(k-\lceil\frac{k}{2}\rceil)}\min\{l(z_j), k-l(z_j)\}. \tag{20.10}$$

Here, k denotes the number of base classifiers $\mathbb{C}$ in the ensemble, m is the number of samples in training dataset, $l(z_j)$ denotes the number of base classifiers that recognizes a sample z_j correctly and $\lceil\frac{k}{2}\rceil$ denotes the votes from base classifiers with same class label.

Objective 2 ***Diversity:*** *Given an ensemble combination* $\mathbb{E}$ *with* k *number of base classifiers, the diversity as per Eq.* (20.10) *is defined as*

$$(\textbf{Maximize}) \quad Obj_{div}(\mathbb{E}) : Diversity\,(\mathbb{E})\,, \tag{20.11}$$

where the $Diversity$ *score is calculated on the training portion* $\mathbb{T}_s$ *of the dataset.*

An ensemble will be assessed for the entropy score attained by the combination of base classifiers. Hence, the optimization of Obj_{div} function will also be a maximization problem.

Ensemble Size: In multi-objective optimization of ensemble classifiers, many researchers used ensemble size (the number of base classifiers in the ensemble combination) as an objective value [1, 8, 9, 23] to minimize it (ensemble size or cardinality). The objective function for minimizing size is defined as:

Objective 3 ***Size:*** *Given an ensemble combinations* $\mathbb{E}$ *and a value* k *identifying the length of the binary string that represents the combination of base classifiers, the objective function for the ensemble size is written as:*

$$(\textbf{Minimize}) \quad Obj_{sz}(\mathbb{E}) : |\mathbb{E}| \mapsto \sum_{i=1}^{k}\mathbb{E}_i\,[\forall_i, \mathbb{E}_i = 1] \qquad subject\ to \quad |\mathbb{E}| \geq 2, \tag{20.12}$$

where the ensemble size is mapped into the number of elements $\mathbb{E}_i$ *having non-zero values in the respective position in the string of base*

classifiers. To be considered valid, a solution needs to have at least two base classifiers.

This optimization of Obj_{sz} is a minimizing task which will reduce the size of the ensemble.

20.5.2 The MO-EoC Framework

A multi-objective evolutionary algorithm ($MOEA$) deals with mathematical optimization problems which require the optimization of more than one criterion simultaneously. MOEAs have been applied successfully in many areas where optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. Unlike in single-objective optimization, MOEA provides a number of Pareto-optimal solutions. As previously noted, a solution is called non-dominated if it is impossible to improve one objective value without degrading some other objective value. For this reason, all Pareto-optimal solutions are treated as equally good. However, a final solution is selected from the Pareto-optimal solutions depending on the subjective preference (trade-offs in satisfying different objectives) of the decision maker.

20.5.2.1 NSGA-II

Among many multi-objective optimization algorithms, we have chosen the widely used NSGA-II (Non-dominated Sorting Genetic Algorithm II) [7] which is an upgraded version of the NSGA algorithm [28]. The NSGA-II MOEA poses several advantages over its predecessor. It improves the NSGA by adopting a fast non-dominated sorting approach and a crowded comparison operator which helps the algorithm to run faster, facilitates the application of the elitist principle, emphasizes non-dominating solutions, maintains population diversity and converges to solutions close to the Pareto-optimal solutions. Hence, the NSGA-II algorithm can simultaneously handle the optimization of multiple objective functions and represents one of the leading Pareto-optimal solution algorithms. It has been successfully applied in various optimization settings, including recently in resource allocation [22], data classification [21], biomarker discovery [30] and biological network analysis [32].

Algorithm 1 provides a pseudocode of NSGA-II. At first a parent population Pop containing Pop_{sz} individuals is randomly initialized. Each individual encodes the problem with a string of size Prb_{sz}. Then, genetic variation operations of *selection*, *recombination* and *mutation* are applied to generate offspring to enter the population ($ChPop$) with the size of Pop_{sz}. The algorithm starts by merging both populations into one population ($MPop$). Then, the combined population is subjected to a sorting procedure with `FastNondominatedSort` function (as shown in Algorithm 2 in the Appendix) to identify non-dominated solutions. The

SortByRankAndDistance function aligns the population into a hierarchy of non-dominated Pareto fronts, assigning the best rank to a solution which is not dominated by any other solutions. The average distance between individuals in each front is calculated using CrowdingDistanceAssignment described in Algorithm 4 (in the Appendix). Then, a function for discriminating individuals in the population is used according to their rank which preserved the elitism. The SelectParentsByRankAndDistance is used for ordering Pareto-front solutions first by their dominance precedence and then by the distance within the front. Then, the new population of offspring is generated after applying genetic variation operators (recombination and mutation). These procedures are repeated until the stopping criteria are satisfied.

Algorithm 1: Pseudocode of NSGA-II algorithm

Input: Pop_{sz}, Prb_{sz}, R_χ, R_μ
Output: $ChPop$

$Pop \leftarrow$ InitializePopulation(Pop_{sz}, Prb_{sz})
EvaluateAgainstObjectiveFunctions(Pop)
FastNondominatedSort(Pop)
$SPop \leftarrow$ SelectParentsByRank(Pop, Pop_{sz})
$ChPop \leftarrow$ RecombinationAndMutation($SPop$, R_χ, R_μ)
while ¬StopCondition **do**
    EvaluateAgainstObjectiveFunctions($ChPop$)
    $MPop \leftarrow$ Merge(Pop, $ChPop$)
    $\mathbb{F} \leftarrow$ FastNondominatedSort($MPop$)
    Parents $\leftarrow \emptyset$
    $F_L \leftarrow \emptyset$
    foreach $F_i \in \mathbb{F}$ **do**
        CrowdingDistanceAssignment(F_i)
        if Size(Parents) + Size(F_i) $> Pop_{sz}$ **then**
            $F_L \leftarrow i$
            Break()
        else
            Parents $\leftarrow$ Merge(Parents, F_i)
        end if
    end foreach
    if Size(Parents) $< Pop_{sz}$ **then**
        $F_L \leftarrow$ SortByRankAndDistance(F_L)
        for P_1 **to** $P_{Pop_{sz} - \text{Size}(F_L)}$ **do**
            Parents $\leftarrow Pi$
        end for
    end if
    $SPop \leftarrow$ SelectParentsByRankAndDistance(Parents, Pop_{sz})
    $Pop \leftarrow ChPop$
    $ChPop \leftarrow$ RecombinationAndMutation($SPop$, R_χ, R_μ)
end while
return $ChPop$

Algorithm 2 shows how we transform the population into non-dominate sorting population. The crowding distance of the non-dominating solutions and their rank are calculated and preserved (lines 4–13). The ranking process of each solution (shown in lines 8–10) in the population is calculated based on their dominance depth. Solutions in the population get the value of *rank* and *crowdingDistance* attributes.

The process of `CrowdingDistanceAssignment` is shown in Algorithm 4 (also in the Appendix). The calculation of crowding distance is restricted by the size of the non-dominated solution front. It returns a positive infinity value for front size ≤ 3 (lines 2–4). Otherwise, we initialize each solution in the fronts with zero distance. For each objective, the crowding distance is calculated for the front (lines 9–20). Here, the front ($\mathbb{F}$) is sorted according to the objective value. For each solution in the front, we update the crowding distance based on its neighbourhood (lines 16–19). Generally, solutions which are far away (not crowded) from other solutions are given a higher rank. The ranking is produced this way to generate a diverse solution set. The procedure returns the front with the associated crowding distances based on the objectives.

Finally, the parent selection method based on the rank and distance is shown in Algorithm 3 (in the Appendix). The parent selection process starts with a randomly selected solution. Then it is compared against other $arity$ sized randomly selected candidates (lines 3–7) for rank and distance. The winning solution is preserved in the population of parents. The process is repeated until the number of parents has reached the arity value.

20.5.2.2 The MO-EoC Using NSGA-II

In our design, we used 28 base classifiers (listed in Table 20.1) from the WEKA data mining software suite [13] to create the ensemble combinations. The method named `EvaluateAgainstObjectiveFunctions` evaluates each pair of objectives listed in Sect. 20.5.1.1. We use a binary string to represent a solution and to encode a pair of objectives inside the multi-objective optimization algorithm. The NSGA-II implementation is taken from the `MOEA Framework` [12], version 2.7, available online.[2] We use the binary presentation of the solution with `Half-uniform-Recombination` (HUX) and `BitFlip` (BF) mutation operators as variation functions of the NSGA-II algorithm. In the case of the HUX recombination operation, half of the non-matching bits are swapped in between the two parents. The probability rate of mutation (R_μ) and recombination (R_χ) is kept unchanged from the default value in the MOEA Framework. Table 20.2 shows the parameter values used for the execution of the MO-EoC algorithm.

[2] http://moeaframework.org.

Table 20.2 Parameter settings of the proposed MO-EoC using NSGA-II

Parameter	Value
Individual type	Binary string
Individual length	28
Population size	100
Maximum evaluation	10,000
Recombination strategy	HUX
Recombination rate (R_χ)	0.75
Mutation strategy	BF
Mutation rate (R_μ)	0.10
A pair of objectives	$\{ Obj_{mcc}, Obj_{div}, Obj_{sz}\}$

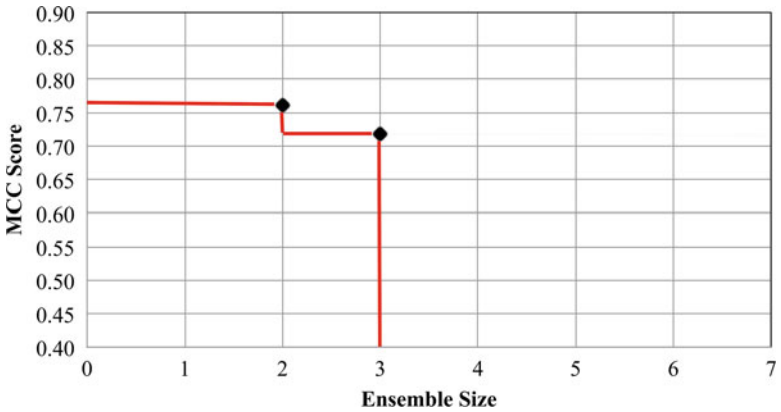

Fig. 20.3 Pareto optimal solutions for optimizing objective pairs of (MCC, Size) for Churn dataset

20.5.3 Computational Results

In each iteration of the NSGA-II algorithm, objectives are taken in pairs from $\{Obj_{mcc}, Obj_{div}, Obj_{sz}\}$ and each pair is evaluated for the Pareto front composed of non-dominated solutions. Then the goodness of those solutions is evaluated to decide the best objective pairs to use in the MO-EoC.

20.5.3.1 MCC vs Size

We evaluate the objective pairs (Obj_{mcc}, Obj_{size}) for our Churn detection dataset by first plotting the values of objective pairs for each of the Pareto-optimal solutions for the datasets as shown in Fig. 20.3. Here, the X-axis identifies the ensemble size and Y-axis identifies the MCC score of the non-dominating solution.

Figure 20.3 shows the scatter plot of objective scores pair (Obj_{mcc}, Obj_{size}) for *Churn* dataset. Here, for 30 independent runs, the experiment reveals only two solutions. One is for an ensemble size of 2 (with an MCC score of 0.72) and the other is for an ensemble of size 3 (with an MCC score of 0.76).

For this experiment, we found that the ensemble size and MCC score are not conflicting objectives and pairing them together did not produce any advantages for enhancing the MCC scores. No study has previously used MCC scores as a performance measure in evaluating the multi-objective ensemble of classifiers. We observe that in our pairwise optimization of the MCC score and ensemble size, the change of the MCC score for increasing the ensemble size is always parallel to the horizontal axis. Hence, the MCC score optimization of (Obj_{mcc}, Obj_{size}) does not demonstrate a conflict between the MCC objective and adhere to the conflicting objective of ensemble size objective in the multi-objective optimization.

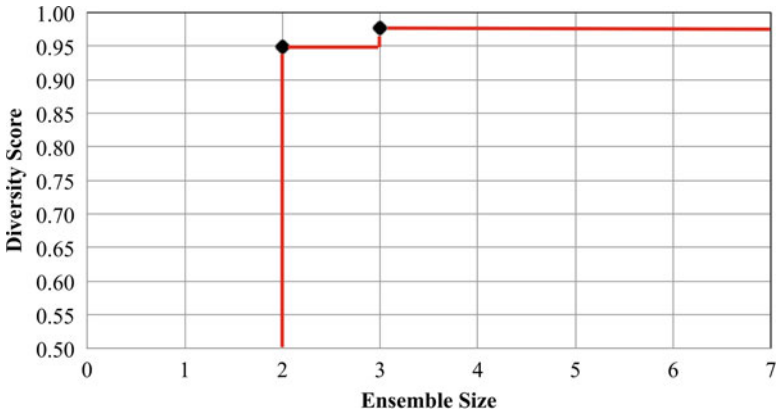

Fig. 20.4 Pareto optimal solutions for optimizing objective pairs of (Div, Size) for Churn dataset

20.5.3.2 Diversity vs Size

Now, we will evaluate the objective pairs (Obj_{div}, Obj_{size}) for the *Churn* dataset. We plot the Pareto-optimal solutions for the optimization of objective pairs in Fig. 20.4. Here, the X-axis identifies the ensemble size and the Y-axis identifies the diversity value of the non-dominating solutions.

Figure 20.4 shows the scatter plot of objective scores pair (Obj_{div}, Obj_{size}) for the churn dataset. Here, all solutions found in 30 repetition grouped into the ensemble of size 2 and 3 with two different diversity scores. The size three ensembles produced slightly better diversity scores than that of size two ensembles. Hence, the outcome has mostly been biased by the minimization of the ensemble size. The disagreement between the base classifiers decision in the solution ensemble with lower size will have more diversity score in compared with

the disagreement by a single base classifier while in large ensemble. Aksela and Laaksonen [2] pointed out that pairwise optimization of diversity and ensemble size always leads to minimize the number of base classifiers. Our result also affirms their observation. Moreover, the optimization of (Obj_{div}, Obj_{size}) pair does not lead to a better generalization (MCC score of 0.586 and 0.598 for solution with 2 and 3 base classifiers, respectively). Therefore, the diversity or the ensemble size without pairing with an objective related to the performance measure will not lead to a better classification outcome.

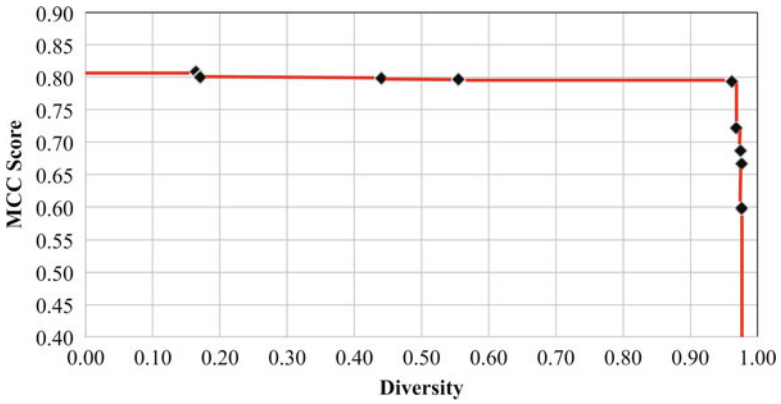

Fig. 20.5 Pareto optimal solutions for optimizing objective pairs of (MCC, Div) for Churn dataset

20.5.3.3 MCC vs Diversity

Now, we evaluate the objective pairs (Obj_{mcc}, Obj_{div}) for the *Churn* dataset. We plot the values of objective pairs for the Pareto-optimal solutions in Fig. 20.5. The X-axis identifies the Diversity score and Y-axis identifies the MCC scores achieved by non-dominating solutions.

Figure 20.5 shows the objective scores for optimizing the pair (Obj_{mcc}, Obj_{div}) in *Churn* dataset. As can be seen from the graph, similar MCC scores (in the range of 0.79–0.80) have been achieved for solutions with different diversity scores. Additionally, solutions with similar diversity scores (in the range of 0.96–0.97) have attained different MCC scores. It is also noticeable that, for Churn dataset, the diversity score of solutions varies in the certain range. Most importantly, lower diversified solutions (within the range) produced better MCC scores. Therefore, it is more prudent to choose a solution having higher MCC score but lower diversity value in case of solution selection from the Pareto-front.

The proposed method applies the genetic algorithm to evaluate three pairs of objectives in multi-objective settings: The first pair to optimize the MCC and the ensemble size, the second pairs of objectives are the ensemble size and Diversity scores, and the final is to optimize the MCC and diversity scores, respectively. The

experimental outcomes revealed that the ensemble size or diversity without pairing with any performance measure does not have a significant effect on increasing predictive performances of the ensemble of classifiers. It is also observed that the pairwise optimization of the MCC and the diversity is the most suitable objective pair for ensemble combination search in multi-objective settings. So, we only consider the pair (Obj_{mcc}, Obj_{div}) as the best objectives for MO-EoC for classifying the *Churn* dataset.

20.6 Classifying Churners with MO-EoC Algorithm

We present the generalization performance of proposed MO-EoC for churner detection using the multi-objective settings in this section. The objectives used in the experiment are the maximization of both MCC and Diversity scores. The summary of threefold cross-validation experimental outcomes of MO-EoC algorithm in 30 independent runs is shown in Table 20.3.

The summary table shows the median MCC score is 0.79 with average of 0.74 for 30 independent runs. From the standard deviations it is evident that the performance of MO-EoC for churner prediction is stable for different classification measures (such as MCC, accuracy, precision and F-Measures).

Table 20.3 Summary statistics of the performances of MO-EoC algorithm for Churn dataset using threefold cross-validation of 30 independent runs

Statistic	MCC	Accu	Prec	FMeas
Minimum	0.53	89.38	0.89	0.89
First quartile	0.69	92.08	0.92	0.92
Third quartile	0.80	95.26	0.95	0.95
Mean	0.74	93.62	0.94	0.93
Median	0.79	95.17	0.95	0.95
Maximum	0.81	95.50	0.95	0.95
Variance	0.01	4.10	0.00	0.00
Stdev	0.07	2.02	0.02	0.02

The performances are measured in MCC, accuracy, precision and F-measure scores

To depict the distribution of predictive performances of MO-EoC found in 30 independent runs considering different measures, we create box-and-whisker plot in Fig. 20.6. The plot shows the performances considering different measures: (a) MCC, (b) Accuracy, (c) Precision and (d) F-Measure scores. From Fig. 20.6a, we found that 25th percentile of the MCC scores are above 0.7. For the accuracy score in Fig. 20.6b, the 25th percentile value is 92%. The spread of precision (Fig. 20.6c) and F-Measure (Fig. 20.6d) scores are very narrow. All of these results are apparent for the MO-EoC as a consistent method in churner classification.

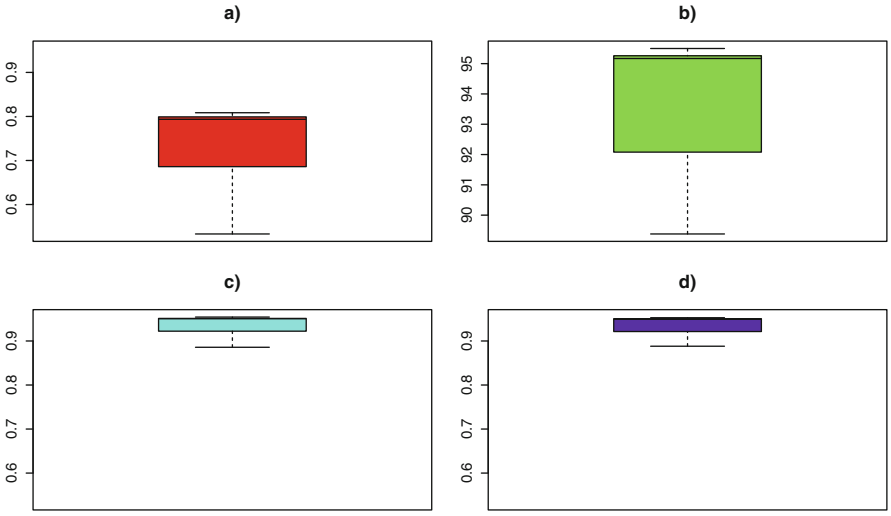

Fig. 20.6 Box-and-whisker plot to show the classification performances of the MO-EoC for Churn dataset using stratified 3-fold cross-validation of 30 independent runs. The performances are measured in (**a**) MCC, (**b**) Accuracy (%), (**c**) Precision and (**d**) F-measure scores

20.6.1 Churner Prediction Performances of Base Classifiers

Table 20.4 shows the classification performances (considering various measures) of base classifiers for threefold cross validation on Churn dataset. Here we report the churner prediction performances of base classifiers for Matthews correlation coefficient (MCC), Accuracy (Accu), Precision (Prec), F-Measure (FMeas), Area Under the Curve (AUC) [20], Sensitivity (SEN) and Specificity (SPEC) scores. The result summary shows the Maximum (Max.), Median and Minimum (Min.) value of each score for 28 base classifiers.

As can be seen from the table, `JRip` classifier attains the maximum MCC score (0.787) among 28 base classifiers. It achieves the best scores for all performance measures except for the AUC and SPEC measures. The `ADTree` classifier achieves the best AUC measure value of 0.899 and the `VFI` classifier achieves the best SPEC score with 0.829.

We plot the generalization performances of 28 base classifiers in Fig. 20.7 for MCC scores. From the graph it is clear that the `JRip` classifier achieves the best score. However, three well-known and widely used classifiers (namely `BFTree`, `REPTree` and `SimpleCart`) are unable to perform well in generalization of churn detection dataset. It indicates that a single best classifier cannot perform better in all

Table 20.4 Classification performances of 28 base classifiers using threefold cross validation on the Churn dataset

Base classifier	MCC	Accu	Prec	FMeas	AUC	SEN	SPEC
BayesNet	0.423	86.35	0.857	0.860	0.837	0.863	0.538
NaiveBayes	0.488	87.91	0.873	0.876	0.832	0.879	0.584
SMO	0.250	86.23	0.833	0.825	0.566	0.862	0.270
SPegasos	0.254	86.23	0.833	0.826	0.569	0.862	0.275
Logistic	0.324	86.41	0.840	0.843	0.806	0.864	0.368
SimpleLogistic	0.199	85.78	0.821	0.815	0.779	0.858	0.238
SGD	0.242	86.05	0.828	0.824	0.566	0.860	0.271
MLPClassifier	0.363	84.49	0.842	0.843	0.769	0.845	0.511
RBFNetwork	0.524	88.99	0.883	0.885	0.844	0.890	0.593
VotedPerceptron	0.118	85.75	0.864	0.795	0.521	0.857	0.161
ADTree	0.660	92.05	0.916	0.918	**0.899**	0.920	0.692
BFTree	0.000	85.51	0.731	0.788	0.500	0.855	0.145
HoeffdingTree	0.488	87.85	0.873	0.876	0.835	0.878	0.587
J48	0.733	93.85	0.936	0.935	0.885	0.938	0.713
LADTree	0.687	92.71	0.923	0.924	0.893	0.927	0.707
REPTree	0.000	85.51	0.731	0.788	0.500	0.855	0.145
Part	0.563	89.68	0.892	0.894	0.806	0.897	0.635
DecisionStump	0.294	85.87	0.831	0.836	0.601	0.859	0.352
FT	0.565	89.89	0.893	0.895	0.810	0.899	0.623
LMT	0.704	93.22	0.929	0.928	0.886	0.932	0.688
RandomTree	0.280	86.59	0.841	0.831	0.613	0.866	0.286
SimpleCart	0.000	85.51	0.731	0.788	0.500	0.855	0.145
IBk	0.214	83.05	0.807	0.816	0.591	0.830	0.351
ConjunctiveRule	0.374	86.68	0.848	0.853	0.770	0.867	0.436
JRip	**0.787**	**94.99**	**0.948**	**0.948**	0.863	**0.950**	0.778
Ridor	0.743	94.09	0.939	0.937	0.824	0.941	0.706
DecisionTable	0.530	89.02	0.884	0.886	0.844	0.890	0.605
VFI	0.023	18.51	0.782	0.123	0.681	0.185	**0.829**
Max	0.787	94.99	0.948	0.948	0.899	0.950	0.829
Median	0.368	86.50	0.853	0.848	0.792	0.865	0.525
Min	0.000	18.51	0.731	0.123	0.500	0.185	0.145

The best results for each of the metrics are shown in boldface

cases but for some cases. So, there is a need of the quest for a classification system that performs consistently well for all cases.

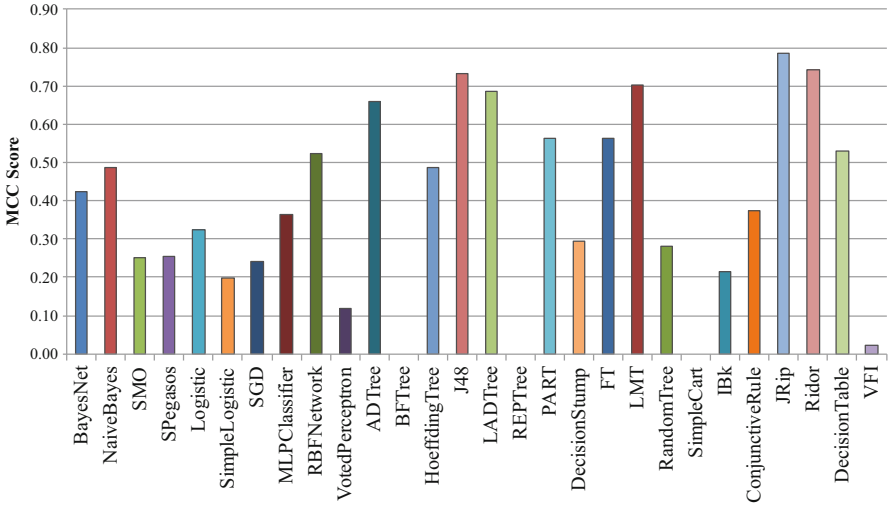

Fig. 20.7 Generalization performances of base classifiers on threefold cross-validation on Churn dataset for MCC scores

20.6.2 Churner Prediction Performances of State-of-the-Art Ensemble of Classifiers

Now we compare the classification performances of state-of-the-art ensemble of classifier algorithms available in WEKA software package. We examined the performance of `AdaBoostM1`, `Bagging`, `RandomForest`, `RandomCommittee` and `Stacking` ensemble of classifiers from WEKA. For comparing the performance of MO-EoC with those state-of-the-art ensemble methods, we used the average performance achieved by our approach. We kept the default parameters unchanged for those ensemble classifiers from the WEKA framework to make the results reproducible.

Table 20.5 shows the comparison of churn prediction performances using various measures for different ensemble of classifiers. It can be seen from the table that the `AdaBoostM1` performed best among state-of-the-art ensemble methods with 0.310 MCC score while considering the MCC measure. Considering the Accuracy, the `RandomCommittee` achieves the best accuracy of 86.44%. In terms of precision measure, `RandomForest` is the best performing ensemble. Finally, considering the F-Measure score, the `AdaBoostM1` opted as the best state-of-the-art ensemble with 0.839 score. Now we consider the average performance of MO-EoC in 30 independent runs and it clearly outperforms all state-of-the-art ensemble methods for all prediction measures used in the experiment.

Table 20.5 Comparison of classification results of MO-EoC (average) and other ensemble of classifiers

Ensemble classifier	MCC	Accu	Prec	FMeas
AdaBoostM1	0.310	86.35	0.837	0.839
Bagging	0.000	85.51	0.731	0.788
RandomForest	0.119	85.75	0.878	0.794
RandomCommittee	0.232	86.44	0.862	0.813
Stacking	0.000	85.51	0.731	0.788
MO-EoC (Avg.)	**0.737**	**93.618**	**0.936**	**0.935**

The best results for each of the metrics are shown in boldface.
All classifiers were tested in the test-set using threefold cross-validation. The performance is measured for the Matthews Correlation Coefficient (MCC), Accuracy (Accu), Precision (Prec) and F-Measure (FMeas) scores

20.6.3 Discussion

In general, the ensemble of classifiers achieves better generalization performances than its associated base classifiers. Now we examine this phenomenon for our proposed MO-EoC. Let us consider the best performances of base classifiers shown in Table 20.4. There we found that the best MCC score 0.787 is achieved by `JRip` which is nearly equal to the median MCC 0.79 of MO-EOC achieved in 30 independent runs. Besides this similarity, the standard deviation for different classification performance measures for MO-EoC is reasonably low (0.07 for MCC, 2% in Accu, 0.02 for both the Prec and F-Measures) for 30 independent runs. These low deviancy and better performances of MO-EoC are evident for its stability and reliability, respectively, in classification of churners from non-churners.

Now we would like to deeply analyse the results in Pareto-front for the optimization of the ensemble combinations by MO-EoC for churn dataset. First, we arbitrarily chose one set of Pareto-optimal solutions from 30 independent runs to observe the training versus testing performance for those solutions. Next we will analyse which base classifiers are appearing in those Pareto-optimal solutions.

In the chosen Pareto-front we find nine Pareto-optimal solutions shown in Table 20.6. Now we see how base classifiers appeared in the ensemble combinations of MO-EoC. We found many base classifiers are not selected in each of the nine ensemble combinations. `BayesNet`, `NaiveBayes`, `SMO`, `Logistics`, `IBk` are some of them. The `FT` and `Ridor` classifiers appear in most of the solutions (in seven out of nine solutions). Most of the Pareto-optimal solutions by MO-EOC are formed with three base classifiers (which are the smallest ensembles in size). The largest ensemble combination (marked as e6) has seven base classifiers. So we find that a small number of base classifiers are selected from 28 available base classifiers to formulate the ensemble combinations in the Pareto-front from MO-EoC.

Now we sort these nine ensemble combinations in order of their diversity score in training and plotted corresponding training MCC score (TrainMCC) with Testing MCC score (TestMCC) in Fig. 20.8. From this observation we found that the

Table 20.6 Variations of base classifiers appeared in nine Pareto-optimal solutions of a run of MO-EoC

Base classifier	e1	e2	e3	e4	e5	e6	e7	e8	e9	Count
BayesNet	0	0	0	0	0	0	0	0	0	0
NaiveBayes	0	0	0	0	0	0	0	0	0	0
SMO	0	0	0	0	0	0	0	0	0	0
SPegasos	0	1	0	0	0	0	1	0	1	3
Logistic	0	0	0	0	0	0	0	0	0	0
SimpleLogistic	0	0	0	0	0	0	0	0	0	0
SGD	0	0	0	0	0	0	0	0	0	0
MLPClassifier	0	0	0	0	0	1	0	0	0	1
RBFNetwork	0	0	0	0	0	0	0	0	0	0
VotedPerceptron	0	0	0	0	0	0	0	0	0	0
ADTree	0	0	0	0	0	0	0	0	0	0
BFTree	0	0	0	0	0	0	0	0	0	0
HoeffdingTree	0	0	0	0	0	0	0	0	0	0
J48	0	0	0	1	0	1	1	0	0	3
LADTree	0	0	0	0	0	0	0	0	0	0
REPTree	0	0	0	0	0	0	0	0	0	0
Part	0	0	0	0	0	0	0	0	0	0
DecisionStump	0	0	0	0	0	0	0	0	0	0
FT	0	1	1	1	1	1	1	0	1	7
LMT	1	0	0	0	1	1	0	1	0	4
RandomTree	0	0	0	0	0	0	0	0	0	0
SimpleCart	0	0	0	0	0	0	0	1	0	1
IBk	0	0	0	0	0	0	0	0	0	0
ConjunctiveRule	0	0	0	0	0	0	0	0	0	0
JRip	1	0	1	1	0	1	0	1	0	5
Ridor	1	0	1	1	1	1	0	1	1	7
DecisionTable	0	1	0	0	0	0	0	0	0	1
VFI	0	0	0	1	0	1	0	1	0	3
#Base classifier	**3**	**3**	**3**	**5**	**3**	**7**	**3**	**5**	**3**	

The number of base classifiers selected for each of the solutions is shown in boldface.
Nine solutions are marked as e1–e9 and selection of a base classifier is marked with 1 under the ensemble combination. The 'Count' column shows the number of times a base classifier has appeared in nine ensemble combinations

increment of diversity score has proportional effect with the training performance in MCC score but the testing performances have the opposite relationship. This asymmetric phenomenon found between the training and testing performances in this figure confirms that the MO-EoC is able to generalize the Churn prediction well.

Finally we conclude from the classification performance comparison between base classifiers, other state-of-the-art ensemble of classifiers and proposed MO-EoC for the churn prediction, it is apparent that the MO-EoC performed consistently

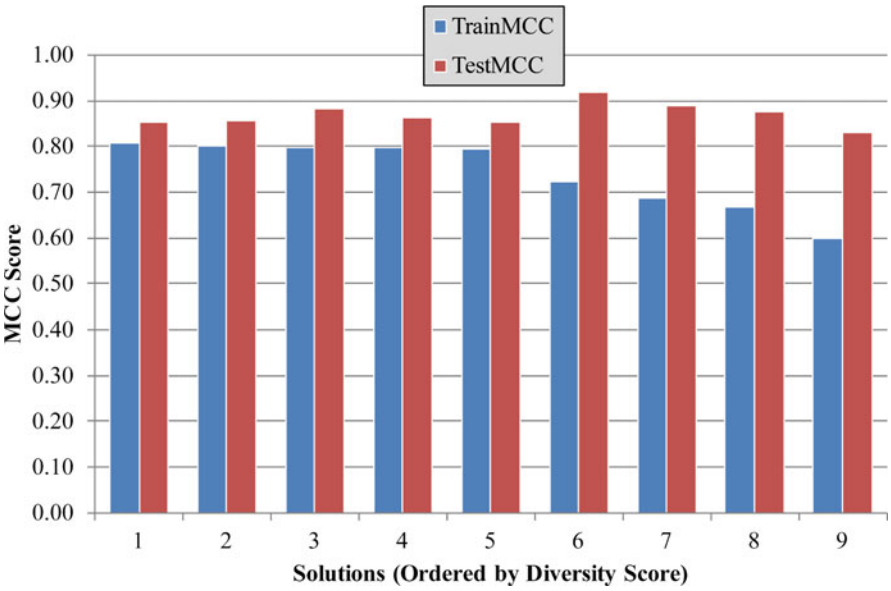

Fig. 20.8 Training and testing performance comparison of Pareto optimal solutions

in 30 independent runs. Moreover, the proposed multi-objective ensemble of classifier system finds ensemble combinations using few base classifiers from large available pool. Hence, the MO-EoC appeared as best method for churn classification compared to its base classifiers and other ensemble methods. We hope that it would perform consistently well for other classification problems in different areas.

20.7 Classifying Churners with Symbolic Regression Analysis

As an additional comparison method, we have used a symbolic regression approach, based on Genetic Programming, to classify and predict the churners from the non-churners. In symbolic regression, the objective is to find a model that fits a given set of input/output data without the user having to explicitly specify the model in advance. For this process we use a software tool called Eureqa from the company called Nutonian,[3] which is a mathematical symbolic regression software tool that has a very simple and useful interface and is free for academic use, thereby making it convenient for verifying the reproducibility of our results. The output of Eureqa is a Pareto optimality curve which shows several models. These models have a 'trade-off' between their capacity to fit the data well and their level of complexity. Eureqa has an ad hoc way of calculating the 'complexity' referred to as the 'size' of the

[3] www.nutonian.com.

models. A weighted function is calculated based on the number and the type of basic 'building blocks' that appear in model equation. Simple 'building blocks' such as addition, subtraction or the operation of introducing an input variable are seen as not very 'complex' and therefore the functions that use only a few of these arithmetic actions will have a relatively low 'complexity' score. Naturally, the opposite is true for more 'complex' building blocks such as quadratic functions. Further explanation of the 'building blocks' in Eureqa and how these have previously been selected by the authors can be found in [6].

Furthermore, the user needs to select the error metric that Eureqa will use as a guiding function to 'evolve' the model (which in turn will relate to the fitness functions used by the Genetic Programming approach). Consistent with their guidelines for binary classification tasks,[4] we used the 'Area under ROC Error' (AUC) as our guiding function, as we are indeed modelling a binary variable (churn or non-churn). To ensure that we will obtain a zero or one result, we 'force' a logistic function on the model. Finally, we select the row weight to equal the value $\frac{1}{occurrences(ChurnResult)}$ as the ratio between churners and non-churners is highly uneven. This step biases the evolutionary approach to consider the churners (of which there are very few samples) as they would provide a higher pay-off than predicting non-churners. This is a common and recommended practice to avoid having a very low number of True Positive predictions.

The threefold datasets were split at random into Training and Test sets. We obtain a Pareto frontier and then we select several models that we have found using only the samples from the training set and test their performance using the samples in the independent test set. For these models, we computed the same metrics (MCC and Accuracy) and we also give the 'Actual Churners Rate' to help us compare these results against the Multi-objective method described earlier in this chapter.

As stated, the Pareto optimality front trades off complexity for minimizing error (such a number is not user defined). To show the progression along this front, and for the purpose of comparison, we used four different models: the least performing model of the front (typically a function of just one variable), the best linear model (as the authors have previously done in [6]), the best model in which the variables still have a 100% positive or 100% negative (or both) variable sensitivity effect on the outcome variable (churn) and finally, the best (and therefore likely the most complex) model presented in the Pareto front.

For each fold, these four models are then tested in the equivalent test sets. To do this step we have to resort to a technique we have developed to address this issue. We first note that we used the AUC error metric to obtain the models in the Pareto frontier; this means that now we need a threshold value to classify the dataset into two distinct classes and evaluate its MCC and Accuracy. This threshold is model dependent and, unfortunately, Eureqa does not give this value to its users (although the value is expected to be provided in future releases as other users have already requested it too).

[4] http://formulize.nutonian.com/documentation/eureqa/tutorials/modeling-binary-outputs/.

We have reverse-engineered this threshold value using, again, Eureqa's functionality. We recall that we followed Eureqa's advice on the training set and evolved the argument of a logistic function. We will call this the 'argument function'. It is then evaluated on the test data, thus providing a value for each of the samples in the test data. We then use Eureqa on a task of fitting churn using these values and only the basic arithmetic actions of subtraction and addition of a constant and introduction of a variable while 'forcing' a logistic function over the whole equation which means we get a model that includes only one constant. The value thus obtained represents the threshold value that we need to use for the outcome of that model. This process was repeated for each of the twelve instances until we had rounded binary results for each that we could compare to the actual churn result in each of the test sets. After obtaining the classification (prediction) results, a confusion matrix is again used to calculate the Accuracy, the 'Actual Churner's Rate' and Matthews Correlation Coefficient (MCC) metrics and compare these to the other methods. This gives us a clear indication of which classification and churn prediction approach is of better quality.

Finally, unlike with the MO-EoC method, the symbolic regression analysis process provides the user with models and shows which variables are used in the prediction models. Therefore, we will conduct various ad hoc inspections to provide a greater understanding and interpretation of each of the models found and provide a brief discussion of this.

20.7.1 Classification Using Symbolic Regression Modelling

As explained in Sect.20.7, we used a symbolic regression modelling tool (Eureqa) to compare the MO-EoC method in terms of churn classification and churn prediction. The results of the evaluation metrics of this method are shown in Table 20.7.

The best results from the symbolic regression modelling process among the 12 instances are shown in bold.

20.7.2 Interpreting Symbolic Regression Results

Although it does not prove to be the most accurate method in terms of ACC and MCC metrics, one of the benefits of symbolic regression for churn prediction is the fact that the actual models and variables (features) used are identified and relationships between these features can be investigated to provide business managers with useful insights. Table 20.8 shows the top five most used features by Eureqa in each of the three folds. This shows us those variables that clearly have a strong relationship with our objective variable: churn.

In Table 20.8 we can see that the variable 'Customer Service Calls' is the most frequently occurring variable in all three folds, meaning it likely has a strong relationship with customer churn behaviour. Furthermore, whether or not the

Table 20.7 Classification results of Eureqa models in all three folds

Classification model	Accu	ACR	MCC
Fold 1 worst Eureqa model	0.689	0.311	0.052
Fold 1 best linear model	0.820	0.789	0.491
Fold 1 model with most 100% variable sensitivities	0.820	0.789	0.491
Fold 1 best Eureqa model	**0.890**	**0.764**	**0.611**
Fold 2 worst Eureqa model	0.590	0.171	0.062
Fold 2 best linear model	0.773	0.714	0.386
Fold 2 model with most 100% variable sensitivities	**0.865**	**0.876**	**0.606**
Fold 2 best Eureqa model	0.867	0.814	0.582
Fold 3 worst Eureqa model	0.572	0.584	0.108
Fold 3 best linear model	0.752	0.689	0.346
Fold 3 model with most 100% variable sensitivities	0.860	0.559	0.456
Fold 3 best Eureqa model	**0.869**	**0.789**	**0.575**

All four models for each fold were tested in the test set and the Accuracy (Accu), the 'Actual Churner's Rate' (ACR) and Matthews Correlation Coefficient (MCC) metrics were computed. The best results when taking all three metrics into account are shown in boldface

Table 20.8 The five most used variables by Eureqa for each fold

Fold 1	Fold 2	Fold 3
CustServ.Calls	CustServ.Calls	CustServ.Calls
Intl.Plan	Intl.Plan	Day.Mins
Day.Mins	Day.Charge	Intl.Plan
Eve.Mins	Eve.Charge	Eve.Mins
Intl.Charge	Intl.Charge	Intl.Charge

customer has an 'International Plan' comes in second place in two, and third place in one of the folds for being the most frequently used variable to predict churn. This provides insights to marketing managers in predicting those consumers who will likely churn, as with this information they are more likely to be able to identify them.

However, although it is interesting to know how many times certain features were included in the models, it is more interesting to know in what way they contributed, i.e., by exhibiting either a positive or a negative relationship with churn. Therefore, for each model, we used the variable sensitivity reports to inspect this. Eureqa's definition for the 'sensitivity' metrics it computes is 'The relative impact within this model that a variable has on the target variable'. As we have explained, we use this information to pick one of the models to investigate. For each of the folds, we picked the best model that still had only variables contributed either in a 100% positive or 100% negative way. As stated, we have twelve instances from taking four models from each of the three folds. However, the three 'worst' results were simply taken for evaluation and comparison purposes and do not provide much insight. We can state here that for each of the three folds, the 'worst' Eureqa results simply provided the logistic function of the variable 'Eve.charge'. As Table 20.7 shows, the results of this model were not very successful at predicting churners. Therefore we move on to investigate the other models.

In Tables 20.9, 20.10, 20.11, 20.12, 20.13 and 20.14 we show the variable sensitivity computations for the models of Eureqa. Firstly, in Table 20.9 we can see that the variable 'Day.Charge' has the highest sensitivity metric by far and is 100% positive. This may indicate to business managers that the higher the customer's 'Day Charge' was (for the telecom service), the more likely they are to churn.

Table 20.9 Variable sensitivity analysis for the best linear model of Fold 1

Variable	Sensitivity	% Positive	% Negative
Day.Charge	1.3729	100%	0%
Day.Mins	0.52689	0%	100%
CustServ.Calls	0.30847	100%	0%
Intl.Plan	0.30109	100%	0%

In this case, the best linear model was also the best model with still all 100% positive or negative sensitivities

Table 20.10 shows the sensitivity analysis for the best model of Fold 1. In this model, the variable 'International Plan' (which is a yes/no variable) has the highest sensitivity and only a positive effect on the predicted value of churn. This means that whenever this variable is higher (i.e., 'yes') churn is more likely to be higher (i.e., churner). However, we do note that some of these more complex models are extremely complicated and each variable individually does not predict the outcome. Rather, the models are 'fitted' by Eureqa to get a lower error metric. However, seeing whether the variables have a positive or negative influence on the total prediction of the model still provides insights into which features are more likely to lead to churners.

Table 20.11 provides the sensitivity analysis for the best linear model of Fold 2. As we can see, the feature with the highest sensitivity is 'Day Charge' again, like in Fold 1. Similar to the best model of Fold 1, here 'International Plan' is one of the variables with the highest sensitivities (here, the second one). Again, it has a fully positive effect which strengthens the finding that whether a customer has an international plan or not affects churn the most. If we could isolate this relationship appropriately, the company would be able to draw conclusions such as whether or not their International Plan was pleasing customers or whether it was receiving too much competition from other providers (which may be one of the reasons those consumers churned and left the company). However, again, like we have stated, these variables are inter-related in the complex models found and need to be taken into consideration together.

Table 20.10 Variable sensitivity analysis for the best model of Fold 1

Variable	Sensitivity	% Positive	% Negative
Intl.Plan	0.35172	100%	0%
VMail.Plan	0.32493	0%	100%
CustServ.Calls	0.21816	75%	25%
Eve.Mins	0.18969	100%	0%
Day.Mins	0.18901	82%	18%
Intl.Charge	0.011258	100%	0%

Table 20.11 Variable sensitivity analysis for the best linear model of Fold 2

Variable	Sensitivity	% Positive	% Negative
Day.Charge	0.62149	100%	0%
Intl.Plan	0.47438	100%	0%
CustServ.Calls	0.28836	100%	0%
Eve.Charge	0.24939	100%	0%
Night.Mins	0.13242	100%	0%
Intl.Charge	0.014844	100%	0%

Table 20.12 shows that again 'Day Charge' is the most sensitive variable in the model to predict churn. Also, 'Customer Service Calls' and 'International Plan' are again in the top three, with a mostly positive effect on the model outcome.

Next is the best linear model of Fold 3 shown in Table 20.13. Here a new variable has the highest 'sensitivity' value 'Day Minutes' (0.934 sensitivity) and is 100% positive. There are three other variables in this model all with much smaller sensitivity values.

Finally, Table 20.14 shows the sensitivity analysis for the best model found in Fold 3. Again 'Day Minutes' has the highest 'sensitivity' value, but this time with a split % of positive and negative influence on the model (54/46%, respectively). A mixture of other variables appears all with a 100% positive influence in the model. Interestingly, 'International Plan' and 'Customer Service Calls' have appeared frequently in the other models we have presented for predicting churn.

Table 20.12 Variable sensitivity analysis for the best model of Fold 2

Variable	Sensitivity	% Positive	% Negative
Day.Charge	0.50578	100%	0%
CustServ.Calls	0.47706	83%	17%
Intl.Plan	0.46105	100%	0%
Eve.Charge	0.25111	100%	0%
Night.Mins	0.13483	100%	0%
Intl.Charge	0.022098	100%	0%

Table 20.13 Variable sensitivity analysis for the best linear model of Fold 3

Variable	Sensitivity	% Positive	% Negative
Day.Mins	0.93535	100%	0%
Intl.Plan	0.26789	100%	0%
CustServ.Calls	0.26688	100%	0%
Intl.Mins	0.0043791	100%	0%

Seeing that several variables are repeatedly some of the most used and most positively 'sensitive' variables inside those models provides us with information to generate useful insights for churn prediction. We now know that some of these variables (e.g., Customer Service Calls, International Plan and Day Minutes) may

Table 20.14 Variable sensitivity analysis for the best model of Fold 3

Variable	Sensitivity	% Positive	% Negative
Day.Mins	0.58804	54%	46%
Intl.Plan	0.44497	100%	0%
CustServ.Calls	0.36845	100%	0%
Eve.Mins	0.21085	100%	0%
Night.Mins	0.11237	100%	0%
Intl.Charge	0.030137	100%	0%

play a more active role in predicting customer churn. We also know those variables that are used less and are therefore less contributing factors to successful churn prediction. For instance, 'Account Length', 'Voicemail Message', 'Day Calls', 'Evening Calls' and a few other variables are not used in any of the models in each of the threefolds. Although these may be important characteristics of the telecom service, in the predictive models we have found here, they are not needed to predict churners. This information could save time and effort for managers as they know where to place emphasis on pleasing customers, i.e., emphasizing those variables that are positively related to churn behaviour rather than those variables that do not affect churn. Besides investigating each of these variables, it shows that Eureqa is advantageous for churn prediction in terms of generating useful insights that can be turned into actions by managers.

20.8 Conclusions

In this chapter we have presented an ensemble learning method for classification and prediction. We have illustrated the effectiveness of this method on a customer churn dataset for the goal of predicting those customers who are going to churn. We have found good results in terms of accurate classification prediction with our evolutionary algorithm MO-EoC. Our approach appears to be the best method for churn classification compared to its base classifiers and other ensemble methods we have examined. Clearly, the predictive power of a dataset is directly related to the quality of the information that it provides (i.e., that can identify features relevant/correlated to causal variables that can explain the phenomenon in question). From this perspective, it is not unusual in the social sciences to 'celebrate' when prediction accuracies are above 0.6. In this case, having obtained MCC scores of 0.8 and above, we are very optimistic that the method is an important contribution.

However, one downside of our algorithm was the lack of interpretability it provides for decision makers. Therefore, we also included an analysis of the customer churn dataset using a symbolic regression approach (based on a commercial and academic-available-for-free genetic programming software which allows reproducibility). The analysis using this method obtained lower results in terms of ACC and MCC than the MO-EoC algorithm; however, our ad hoc analyses of the resulting models provide some actionable insights for business decision makers. As

stated in Sect. 20.7.2, the variables 'Customer Service Calls', 'International Plan' and 'Day Minutes' were among the most likely variables to have an active role in predicting customer churn when no segmentation of the customer base is performed.

Finally, we may state that methods like these (the MO-EoC algorithm and symbolic regression analysis) together provide a comprehensive analysis and reliable outcomes for problems such as churn prediction. Accurate methods such as MO-EoC can be used to predict those customers likely to churn, and the further analysis using symbolic regression approaches can provide insight into how and what types of resources companies can invest in to accurately predict churners to prevent them from churning.

Acknowledgements Pablo Moscato acknowledges previous support from the Australian Research Council Future Fellowship FT120100060 and Australian Research Council Discovery Projects DP120102576 and DP140104183.

Appendix

This extended Appendix provides additional algorithms used in this chapter. Each of these algorithms intend to solve some sub-problems dealt in the main algorithm and readers are highly encouraged to investigate these algorithms for themselves for the continued journey and challenge for solving business and consumer analytics using NSGA-II.

Algorithm 2: Pseudocode of FASTNON-DOMINATEDSORT

Input: The population $\mathbb{P}$
Output: Fronts $\mathbb{F}$

foreach $\mathbb{P}_i \in \mathbb{P}$ **do**
  $\mathbb{R}$.Add($\mathbb{P}_i$)
end foreach
$rank \leftarrow 1$
while $\mathbb{R} \neq \phi$ **do**
  $\mathbb{F} \leftarrow \phi$
  foreach $\mathbb{R}_i \in \mathbb{R}$ **do**
    $\mathbb{F}$.Add($\mathbb{R}_i$)
  end foreach
  foreach $F_i \in \mathbb{F}$ **do**
    $\mathbb{R}$.Remove(F_i)
    F_i.setRank($rank$)
  end foreach
  $\mathbb{F} \leftarrow$ CrowdingDistanceAssignment($\mathbb{F}$)
  $rank \leftarrow rank + 1$
  $\mathbb{P}$.Add($\mathbb{F}$)
end while
return $\mathbb{P}$;

Algorithm 3: Pseudocode of SELECTPARENTSBYRANKANDDISTANCE

Input: Population $\mathbb{P}$, Tournament Size $arity$
Output: Parent Population Parents

```
1  for si ← 1 :arity do
2      P_win ← getRandomSoln(P)
3      for i ← 1 : Size(arity) do
4          P_cand ← getRandomSoln(P)
5          flag ← CompareRankDist(P_win, P_cand)
6          if flag > 0 then
7              P_win ← P_cand
8          end if
9      end for
10     Parents_si ← P_win
11 end for
12 return Parents;
```

Algorithm 4: Pseudocode of CROWDINGDISTANCEASSIGNMENT

Input: Input Fronts $\mathbb{F}$
Output: Output Fronts $\mathbb{F}$

```
1  n ← Size(F)
2  if n ≤ 3 then
3      foreach F_i ∈ F do
4          F_i.setDistance(+∞)
5      end foreach
6  else
7      nObj ← getNumOfObjs(F_1)
8      foreach F_i ∈ F do
9          F_i.setDistance(0)
10     end foreach
11     foreach Obj_i ∈ nObj do
12         F.Sort(Obj_i)
13         Obj_min ← F_1.getObj(Obj_i)
14         Obj_max ← F_n.getObj(Obj_i)
15         F_1.setDistance(+∞)
16         F_n.setDistance(+∞)
17         foreach F_i ∈ F do
18             Dist_cr ← F_i
19             Dist_Range ← (Obj_max − Obj_min)
20             Dist_nbr ← (F_{i+1}.getObj(Obj_i) − F_{i−1}.getObj(Obj_i))
21             Dist_cr ← Dist_cr + (Dist_nbr / Dist_Range)
22             F_i.setDistance(Dist_cr)
23         end foreach
24     end foreach
25 end if
26 return F;
```

References

1. K. Ahmadian, A. Golestani, M. Analoui, and M.R. Jahed. Evolving ensemble of classifiers in low-dimensional spaces using multi-objective evolutionary approach. In *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on*, pages 217–222, July 2007.
2. Matti Aksela and Jorma Laaksonen. Using diversity of errors for selecting members of a committee classifier. *Pattern Recognition*, 39(4):608–623, April 2006.
3. Massimiliano Caramia and Paolo Dell'Olmo. *Multi-objective Management in Freight Logistics: Increasing Capacity, Service Level and Safety with Optimization Algorithms*, chapter Multi-objective Optimization, pages 11–36. Springer London, London, 2008.
4. Arjun Chandra and Xin Yao. Ensemble learning using multi-objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms*, 5(4):417–445, 2006.
5. Chien-Yuan Chiu and B. Verma. Multi-objective evolutionary algorithm based optimization of neural network ensemble classifier. In *Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on*, pages 1–5, Dec 2014.
6. Natalie Jane de Vries, Jamie Carlson, and Pablo Moscato. A data-driven approach to reverse engineering customer engagement models: Towards functional constructs. *PLoS ONE*, 9(7):e102768, 2014.
7. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.
8. E.M. Dos Santos, R. Sabourin, and P. Maupin. Single and Multi-Objective Genetic Algorithms for the Selection of Ensemble of Classifiers. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 3070–3077, 2006.
9. Eulanda M. Dos Santos, Robert Sabourin, and Patrick Maupin. Pareto analysis for the selection of classifier ensembles. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, pages 681–688, New York, NY, USA, 2008. ACM.
10. R. Dutt and A.K. Madan. Predicting biological activity: Computational approach using novel distance based molecular descriptors. *Computers in Biology and Medicine*, 42(10):1026–1041, 2012.
11. Shenkai Gu, Ran Cheng, and Yaochu Jin. Multi-objective ensemble generation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):234–245, 2015.
12. David Hadka. MOEA Framework: A Free and Open Source Java Framework for Multiobjective Optimization, 2014.
13. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
14. M. N. Haque, M. N. Noman, R. Berretta, and P. Moscato. Optimising weights for heterogeneous ensemble of classifiers with differential evolution. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 233–240, July 2016.
15. Mohammad Nazmul Haque, Nasimul Noman, Regina Berretta, and Pablo Moscato. Heterogeneous ensemble combination search using genetic algorithm for class imbalanced data classification. *PLoS ONE*, 11(1):e0146116, 01, 2016.
16. Yaochu Jin and B. Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):397–415, May 2008.
17. Giuseppe Jurman, Samantha Riccadonna, and Cesare Furlanello. A comparison of MCC and CEN error measures in multi-class prediction. *PLoS ONE*, 7(8):e41882, 08, 2012.
18. Gulshan Kumar and Krishan Kumar. The Use of Multi-Objective Genetic Algorithm Based Approach to Create Ensemble of ANN for Intrusion Detection. *International Journal of Intelligence Science*, 2(October):115–127, 2012.

19. Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.
20. Charles X Ling, Jin Huang, and Harry Zhang. AUC: a statistically consistent and more discriminating measure than accuracy. In *IJCAI*, volume 3, pages 519–524, 2003.
21. Seema Mane, Shilpa Sonawani, and Sachin Sakhare. Hybrid multi-objective optimization approach for neural network classification using local search. In *Innovations in Computer Science and Engineering*, pages 171–179. Springer, 2016.
22. Anabel Martínez-Vargas, Josué Domínguez-Guerrero, Ángel G Andrade, Roberto Sepúlveda, and Oscar Montiel-Ross. Application of NSGA-II algorithm to the spectrum assignment problem in spectrum sharing networks. *Applied Soft Computing*, 39:188–198, 2016.
23. Tien Thanh Nguyen, A.W.-C. Liew, Xuan Cuong Pham, and Mai Phuong Nguyen. Optimization of ensemble classifier system based on multiple objectives genetic algorithm. In *Machine Learning and Cybernetics (ICMLC), 2014 International Conference on*, volume 1, pages 46–51, July 2014.
24. Ruba Obiedat, Mouhammd Alkasassbeh, Hossam Faris, and Osama Harfoushi. Customer churn prediction using a hybrid genetic programming approach. *Scientific Research and Essays*, 8(27):1289–1295, 2013.
25. A. Rahman and B. Verma. Cluster oriented ensemble classifiers using multi-objective evolutionary algorithm. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–6, Aug 2013.
26. A. Santana, R.G.F. Soares, A.M.P. Canuto, and Marcilio C P de Souto. A dynamic classifier selection method to build ensembles using accuracy and diversity. In *Neural Networks, 2006. SBRN '06. Ninth Brazilian Symposium on*, pages 36–41, Oct 2006.
27. L Shi, G Campbell, WD Jones, F Campagne, S Walker, Z Su, et al. The MAQC-II project: A comprehensive study of common practices for the development and validation of microarray-based predictive models. *Nature biotechnology*, 2010.
28. Nidamarthi Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.
29. Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.
30. Fatemeh Vafaee. Using multi-objective optimization to identify dynamical network biomarkers as early-warning signals of complex diseases. *Scientific Reports*, 6:22023, 2016.
31. Zhi-Hua Zhou and Nan Li. Multi-information ensemble diversity. In *Multiple Classifier Systems: 9th International Workshop, MCS 2010, Cairo, Egypt, April 7–9, 2010. Proceedings*, pages 134–144. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
32. S Zickenrott, VE Angarica, BB Upadhyaya, and A Del Sol. Prediction of disease–gene–drug relationships following a differential network analysis. *Cell Death & Disease*, 7(1):e2040, 2016.