Troika – An Improved Stacking Schema for Classification Tasks

Eitan Menahem, Lior Rokach, Yuval Elovici

Deutsche Telekom Laboratories at Ben-Gurion University
Ben Gurion University, Be'er Sheva, 84105, Israel
{eitanme, liorrk ,elovici}@bgu.ac.il

Abstract

The idea of ensemble methodology is to build a predictive model by integrating multiple models. It is well-known that ensemble methods can be used for improving prediction performance. Researchers from various disciplines such as statistics, machine learning, pattern recognition, and data mining have considered the use of ensemble methodology. Stacking is a general ensemble method in which a number of base classifiers are combined using one meta classifier which learns theirs outputs. The advantage of stacking is that it is simple, in most cases performs similar to the best classifier, and it is capable to combine classifiers induced by different inducers. The disadvantage of stacking is that on multiclass problems, stacking seems to perform worse than other meta-learning approaches. In this paper we present *Troika*, a new method for improving ensemble classifiers using stacking. The new scheme is built from three layers of combining classifiers. The new method was tested on various datasets and the results indicate the superiority of the proposed method to other legacy ensemble schemes, Stacking and StackingC, especially when the classification task consists of more than two classes

Keywords: Machine Learning, Meta Combination, Ensemble of Classifiers, Stacked Generalization.

1. Introduction

A classifier is a classification model which assigns an unclassified instance to a predefined set of classes. The classifier may be induced by using a learning algorithm

(also known as an inducer), such as C4.5 (Quinlan, 1993) or SVM (Boser et. al. 1992; Vapnik 1998). Ensemble methodology considers combining multiple classifiers to work collectively in order to compensate each other's weaknesses and to generate better classifications through some kind of fusion strategy. Many works had been made to investigate new techniques of combining multiple classifiers to produce a single classifier (A very short list: Breiman, 1996c, Aydın et al. 2009, S. Cohen et al. 2007 and G. Peter Zhang 2007), however, in this research we focus our investigation on three stacked generalization methods; Stacking (Wolpert, D., 1992), StackingC (Seewald, 2003) and our novel technique - Troika.

The rest of the paper is organized as follows. In section 2 we present the related work, and focus on two important Stacked-Generalization Meta-classifiers – Stacking, and StackingC. We present some of their weaknesses. In section 3 we present a new ensemble schema called *Troika* which in contrary to Stacking and StackingC it has three layers of combining classifiers. In section 4 we report the results of an experimental study that was performed to determine which scheme performs better. Finally we present the conclusions and discuss open issues for future research.

2. Related Work

Meta-learning is a process of learning from learners (classifiers). The training of a meta-classifier is composed of two or more stages, rather than one stage, as with standard learners. In order to induce a meta classifier, first the base classifiers are trained (stage one), and then the Meta classifier (second stage). In the prediction phase, base classifiers will output their classifications, and then the Meta-classifier(s) will make the final classification (as a function of the base classifiers).

2.1. Stacking

Stacking is a technique whose purpose is to induce which classifiers are reliable and which are not. Stacking is usually employed to combine models built by different inducers. The idea is to create a meta-dataset containing a tuple for each tuple in the original dataset. However, instead of using the original input attributes, it uses the classifications predicted by the classifiers as the input attributes. The target-attribute

remains as in the original training set. A test instance is first classified by each of the base classifiers. These classifications are fed into a meta-level training set from which a meta-classifier is produced.

This classifier (Meta-classifier) combines the different predictions into a final one. It is recommended that the original dataset should be partitioned into two subsets. The first subset is reserved to form the meta-dataset and the second subset is used to build the base-level classifiers. Consequently the meta-classifier predictions reflect the true performance of base-level learning algorithms. Stacking performance can be improved by using output probabilities for every class label from the base-level classifiers. It has been shown that with stacking the ensemble performs (at best) comparably to selecting the best classifier from the ensemble by cross validation (Dzeroski and Zenko, 2004).

Table 1: Original training-set

Attributes	Class
$Attr_{Vec1}$	C_a
$Attr_{Vec2}$	C_b
$Attr_{Vec3}$	C_a
$Attr_{Vec4}$	C_c
.	.
.	.
.	.
$Attr_{Vecn}$	C_b

Table 2: Sample class probability distribution

C_a	C_b	C_c
0.9	0.07	0.03
0.1	*0.85*	0.05
0.8	0.13	0.07
0.3	0.2	*0.5*
.	.	.
.	.	.
.	.	.
0.2	*0.75*	0.05

Table 3: Meta training set, Stacking

$Classifier_1$			$Classifier_2$				$Classifier_n$			Class
a	b	c	a	b	c		a	b	c	
$P_{1,a,1}$	$P_{1,b,1}$	$P_{1,c,1}$	$P_{2,a,1}$	$P_{2,b,1}$	$P_{2,c,1}$	…	$P_{N,a,1}$	$P_{N,b,1}$	$P_{N,c,1}$	C_a
$P_{1,a,2}$	$P_{1,b,2}$	$P_{1,c,2}$	$P_{2,a,2}$	$P_{2,b,2}$	$P_{2,c,2}$	…	$P_{N,a,2}$	$P_{N,b,2}$	$P_{N,c,2}$	C_b
$P_{1,a,3}$	$P_{1,b,3}$	$P_{1,c,3}$	$P_{2,a,3}$	$P_{2,b,3}$	$P_{2,c,3}$	…	$P_{N,a,3}$	$P_{N,b,3}$	$P_{N,c,3}$	C_a
$P_{1,a,4}$	$P_{1,b,4}$	$P_{1,c,4}$	$P_{2,a,4}$	$P_{2,b,4}$	$P_{2,c,4}$	…	$P_{N,a,4}$	$P_{N,b,4}$	$P_{N,c,4}$	C_c
$P_{1,a,n}$	$P_{1,b,n}$	$P_{1,c,n}$	$P_{2,a,n}$	$P_{2,b,n}$	$P_{2,c,n}$	…	$P_{N,a,n}$	$P_{N,b,n}$	$P_{N,c,n}$	C_b

Table 1 shows an example of dataset with three classes (a, b and c) and *n* examples. It shows the original training set with its attribute vectors and class values.

Table 2 shows how a class probability distribution of one sensible classifier may appear. The maximum probabilities are shown in italics and denote the classes which would be predicted for each example. There is one such set of class probability distributions for each base classifier.

Table 3 shows the Meta training set for Stacking with *n* base classifiers which is used to train a Meta classifier that will output the final prediction. $P_{i,j,m}$ denotes the probability given by the base classifier *i* for class *j* on example number *m*. The classes are mapped to an indicator variable such that only class "a" is mapped to 1, and all other classes are mapped to 0. In this example there are, of course, two other such training sets for class b and c which differ only in the last column and are thus not shown.

2.2. StackingC

StackingC is a Stacking variation. In empirical tests Stacking showed significant performance degradation for multi-class datasets. StackingC was designed to address this problem. In StackingC, each base classifier outputs only one class probability prediction (Seewald, 2003). Each base classifier is trained and tested upon one particular class while stacking output probabilities for all classes and from all

component classifiers. Tables 4, show an illustration of StackingC on a dataset with three classes (a, b and c), n examples, and N base classifiers. $P_{i,\ j,\ m}$ refers to the probability given by base classifier i for class j on example number m

Table 4: Meta training set for class$_a$, StackingC

Classifier_1	Classifier_2		Classifier_n	Class=a?
$P_{1,a,1}$	$P_{2,a,1}$	…	$P_{N,a,1}$	1
$P_{1,a,2}$	$P_{2,a,2}$	…	$P_{N,a,2}$	0
$P_{1,a,3}$	$P_{2,a,3}$	…	$P_{N,a,3}$	1
$P_{1,a,4}$	$P_{2,a,4}$	…	$P_{N,a,4}$	0
		…		
$P_{1,a,n}$	$P_{2,a,n}$	…	$P_{N,a,n}$	0

Table 4 shows the corresponding Meta training set for StackingC which consists only of those columns from the original meta training set which are concerned with class = C_a, i.e., $P_{i,\ j\ ,m}$ for all i, j and m. Concerning the Meta classifier's training-set, StackingC's differs from Stacking's not only in the last attribute (the class indicator variable), but also by the amount of attributes; StackingC have fewer attributes by a factor equal to the number of classes. This necessarily leads to more diverse linear models, which Seewald (2003) believes to be one mechanism by which it outperforms Stacking. Another reason may simply be that with fewer attributes, the learning problem becomes easier to solve, provided only irrelevant information is removed. The dimensionality of the Meta dataset is reduced by a factor equal to the number of classes, which leads to faster learning. In comparison to other ensemble learning methods this improves Stacking's advantage further, making it the most successful system by a variety of measures.

StackingC improves on Stacking in terms of significant accuracy differences, accuracy ratios, and runtime. These improvements are more evident for multi-class datasets and have a tendency to become more pronounced as the number of classes increases. StackingC also resolves the weakness of Stacking in the extension proposed by Ting and Witten (1999) and offers a balanced performance on two-class and multi-class datasets.

2.3. Why use Stacking?

Seewald (2003) has shown that all ensemble learning systems, including StackingC (Seewald, 2002), Grading (Seewald and Fuernkranz, 2001) and even Bagging (Breiman, 1996) can be simulated by Stacking (Wolpert, 1992). To do this they give functionally equivalent definitions of most schemes as Meta-classifiers for Stacking. Dzeroski and Zenko (2004) indicated that the combination of SCANN (Merz, 1999), which is a variant of Stacking, and MDT (Ting and Witten, 1999) plus selecting the best base classifier using cross validation seems to perform at about the same level as Stacking with Multi-linear Response (MLR).

2.4. Weaknesses of Stacking

Seewald (2003) presented strong empirical evidence that Stacking in the extension proposed by Ting and Witten (1999) performs worse on multi-class than on two-class datasets, for all but one meta-learner he investigated. The explanation given was that when the dataset has a higher number of classes, the dimensionality of the meta-level data is proportionally increased. This higher dimensionality makes it harder for meta-learners to induce good models, since there are more features to be considered. The increased dimensionality has two more drawbacks. First, it increases the training time of the Meta classifier; in many inducers this problem is acute. Second, it also increases the amount of memory which is used in the process of training. This may lead to insufficient resources, and therefore may limit the number of training cases (instances) from which an inducer may learn, thus damaging the accuracy of the ensemble.

2.5. Weaknesses of StackingC

During the learning phase of StackingC it is essential to use one-against-all class binarization and regression learners for each class model. This class binarization is believed to be a problematic method especially when class distribution is highly non-symmetric. It has been illustrated (Fürnkranz, 2002) that handling many classes is a major problem for the one-against-all binarization technique, possibly because the resulting binary learning problems increasingly skewed class distributions. An alternative to one-against-all class binarization is the one-against-one binarization in which the basic idea is to convert a multiple class problem into a series of two-class problems by training one classifier for each pair of classes, using only training

examples of these two classes and ignoring all others. Assuming a *k*-class problem, a new example is classified by submitting it to each of the $\frac{k(k-1)}{2}$ binary classifiers, and combining their predictions using *k* meta classifiers as described in section 2.2. We have found in our preliminary experiments that this binarization method yields noticeably poor accuracy results when the number of classes in the problem increases. Later, after performing a much wider and broader experiment on StackingC in conjunction with the one-against-one binarization method, we came to this same conclusion. An explanation might be that, as the number of classes in a problem increases, the greater is the chance that any of the $\frac{k(k-1)}{2}$ base classifiers will give a wrong prediction. There are two reasons for this. First, when predicting the class of an instance, only $k-1$ out of $\frac{k(k-1)}{2}$ classifiers may predict correctly. This is because only $k-1$ classifiers were trained on any specific class. We can see that as *k* increases, the percentage of classifiers which may classify correctly is decreasing, and will descend practically to zero:

$$\lim_{k->\infty} \frac{k-1}{\frac{k(k-1)}{2}} = \lim_{k->\infty} \frac{2}{k} = 0 \qquad (1)$$

The second reason is that in one-against-one binarization we use only instances of two classes – the instances of each one of the pair classes, while in one-against-all we use all instances, and thus the number of training instances for each base classifier in one-against-one binarization is much smaller than in the one-against-all binarization method. Thus using the one-against-one binarization method may yield inferior base classifier.

2.6. Converting Multiclass Classification Problems into Binary Classification Problems

There are several alternatives to decompose the multiclass problem into binary subtasks. Lorena and de Carvalho (2007) and S. Cohen et al (2007) survey all popular methods.

The most straightforward method to convert *k* class classification problems into *k*-two class classification problems has been proposed by Anand et al. (1995). Each problem considers the discrimination of one class to the other classes. Lu and Ito (1999) extend Anand's method and propose a new method for manipulating the data based on the class relations among the training data. By using this method, they divide a *k* class classification problem into a series of *k(k-1)/2* two-class problems where each problem considers the discrimination of one class to each one of the other classes. The researchers used neural networks to examine this idea.

A general concept aggregation algorithm called *Error-Correcting Output Coding* (ECOC) uses a code matrix to decompose a multi-class problem into multiple binary problems (Dietterich and Bakiri, 1995). ECOC for multi-class classification hinges on the design of the code matrix.

Sivalingam et al. (2005) propose to transform a multiclass recognition problem into a minimal binary classification problem using the Minimal Classification Method (MCM) aided with error correcting codes. The MCM requires only $log_2 k$ classifications because instead of separating only two classes at each classification, this method separate two groups of multiple classes. Thus the MCM requires small number of classifiers and still provide similar accuracy performance.

Data-driven Error Correcting Output Coding (DECOC) (Zhoua et al., 2008) explores the distribution of data classes and optimizes both the composition and the number of base learners to design an effective and compact code matrix. Specifically, DECOC calculate the confidence score of each base classifier based on the structural information of the training data and use sorted confidence scores to assist the determination of code matrix of ECOC. The results show that the proposed DECOC is able to deliver competitive accuracy compared with other ECOC methods, using parsimonious base learners than the pairwise coupling (one-vs-one) decomposition scheme.

It should be noted that finding new methods for converting multiclass classification problems into binary classification problems is not one of the goals of this paper. Still, we are using in our experimental study three different methods for this conversion.

3. Troika ensemble scheme

A new ensemble methodology, Troika, is designed to address the Stacking and StackingC problems described above. Troika's ensemble scheme is general purpose and can be used to combine any type of base classifiers which were trained on any subgroup of possible classes of a problem's domain. In other words, it is possible with Troika to combine models (base classifiers) that were trained on, and therefore may later predict, non congruent datasets, in terms of instances classes.

The main idea of Troika is to combine base classifier in three stages. In the first stage it will combine all base classifiers[1] using specialist classifiers. The specialists are classifiers which have a dichotomous model[2]; each specialist's (specialist classifier) task is to distinguish between pairs of classes from the problem domain, and no two specialists are alike, i.e., each specialist is specialized in distinguishing between different pairs of classes. We will use the notation Sp_{i-j} to indicate the specialist $<i,j>$. Sp_{i-j}'s task is to output the probabilities that an input instance belongs either to $class_i$ or to $class_j$ [3] via vector of two values: {$P(class_i)$, $P(class_j)$}; since those probabilities are complementary, we will later use only one these of the class with the smallest index. Let k be the number of classes in a problem domain, i= {0…k-2} and j= {i+1…k-1}. The exact specialist classifiers number equals $\binom{k}{2}$, where k is the number of classes in the problem domain. A specialist classifier output, $P_{inst,i-j}$, is the computed probability that an input instance, *inst*, belongs to $class_i$ [4]. Given an instance *inst* belonging to $class_i$ or $class_j$, we will expect Sp_{i-j} to predict the *inst* class correctly most of the time. Conversely, when *inst* real class is *not i* or *j* the output of Sp_{i-j} will certainly be faulty in an unpredicted way. For example, Sp_{2-5} indicates $specialist_{2-5}$ which may distinguish between $class_2$ and $class_5$. If an instance *inst* of $class_0$ is given to Sp_{2-5}, we cannot make a preliminary assumption about Sp_{2-5}'s

[1] Sometimes refer to as Level-0 or base layer

[2] A one-against-one binarization

[3] This raises the question: what if the input instance does not belongs to either $class_i$ or $class_j$? The answer is that Sp_{i-j} will certainly be wrong in its predictions, because $P(class_i)$ and $P(class_j)$ must add to 1, what seemingly suggest that at least one probability will be equal or greater than 0.5 which translates to wrong classification.

[4] Input instance *inst* has a computed probability $(1-P_{inst,i-j})$ of belonging to $class_j$

output[5]. This is why we need to learn the characteristics and patterns of the behavior of specialists to be able to predict when specialists are correct and when they are not. This is exactly what is done in the next stage, the meta-classifier layer.

The second stage is the meta-classifiers layer. This layer's task is to learn the prediction characteristics of the specialist classifiers. The method used to accomplish this task is to combine the specialist classifiers using *k* meta-classifiers. Each meta-classifier is in charge of one class only, and will combine all the specialist classifiers which are able to classify its own class; meta-classifier$_m$[6] will combine all specialists Sp_{i-j} whose $i=m$ or $j=m$. The meta-classifier will compute a probability $P_{inst,c}$ as an output. $P_{inst,c}$ stands for the computed probability that a given input instance *inst* belongs to class$_c$ [7]. The meta-classifiers are trained in a one-against-all fashion, rather than one-against-one as with specialist classifiers. We will explain the logic behind this choice later.

The third stage is the super classifier layer. This layer contains only one classifier: the super classifier. The goal of this stage is to produce Troika's final prediction. The inputs of the super classifier are the outputs $P_{inst,c}$ produced by the meta classifiers in the previous stage. In the training phase, the Super classifier learns the conditions in which one or more of the meta-classifiers predict correctly or incorrectly. The super classifier's output is a vector of probabilities (one value for each class) which forms the final decision of the Troika ensemble scheme.

3.1. Troika Architecture

Figure 1 presents the schematic of the Troika ensemble's architecture. Troika uses three distinguishable layers of combining classifiers, and is arranged in a tandem-like order. An instance of a problem's domain feeds the layer$_0$ classifiers (base classifiers). Layer$_0$ classifiers will output their predictions to layer$_1$'s inputs. Classifiers on layer$_1$ (specialists classifiers) will combine layer$_0$'s classifiers, and then feed their predictions to layer$_2$ inputs. Layer$_2$ classifiers (meta-classifiers) in their

[5] This is because in the training phase Sp_{2-5} had been given only instances of class$_2$ and class$_5$

[6] $m=\{0\ldots k-1\}$

[7] to remind: $c=\{0\ldots k-1\}$

turn, will combine layer$_1$'s classifiers predictions and feed their own predictions to layer$_3$ inputs. The layer$_3$ classifier (super classifier) will combine layer$_2$'s predictions, and ultimately, produce a final prediction.

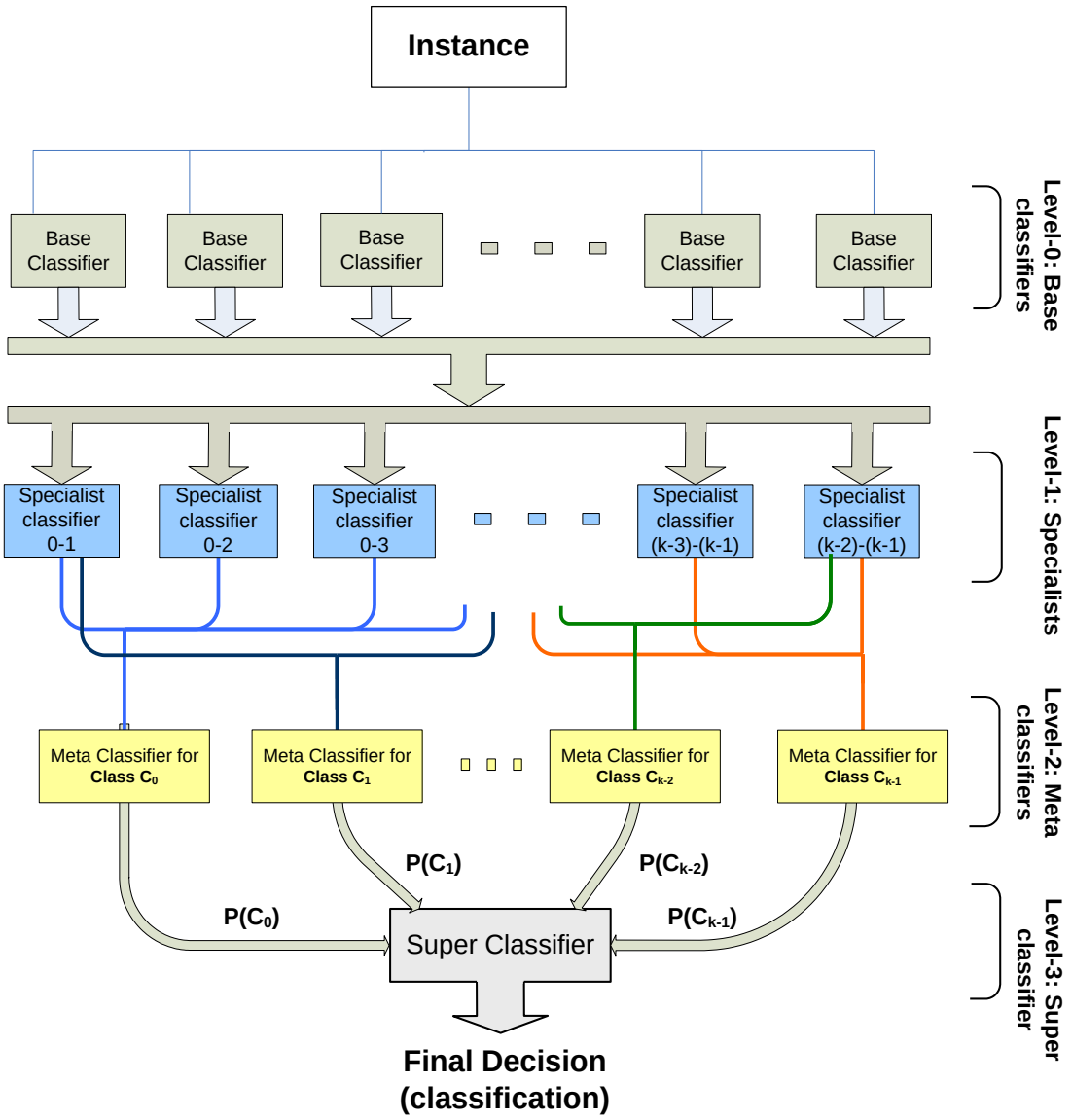


Figure 1: Troika architecture - showing three layers of combiners (level$_1$ to level$_3$) and a base classifier level (level$_0$).

3.2. Why do we need three layers of combining classifiers?

Stacking and StackingC are both using only one layer of combining classifier(s). Isn't this enough? Experiments show it is not. Stacking, which uses one classifier as meta classifier (combiner) suffers from a dimensionality problem. It is not sufficiently

scalable. When increasing the number of classifiers, Stacking usually does not utilize them as well as StackingC will do, producing worse predictions than StackingC in many cases; when increasing classes count in a dataset, Stacking's accuracy deteriorates much more quickly than StackingC's does[8].

StackingC meta-classifiers, on the other hand, are trained in a one-against-all class binarization technique. Using this method, makes StackingC not scalable with classes count in the dataset. This is because the resulting binary learning problems which are input to the meta-classifier have increasingly skewed class distributions. It was mentioned earlier in this paper that one-against-one binarization is an even worse method for training StackingC meta-classifiers.

To escape the dimensionality problem, it is advisable that $layer_1$ will contain more than one combining classifier. To extricate us from the skewed class distributions produced by the one-against-all training method, we will use a one-against-one binarization training method. We choose to call the combiners of $layer_1$ "specialists" because they specialize in differentiating between pairs of classes

As explained before, a large majority of $layer_1$ combiners will always predict falsely and it is for this reason alone that another layer of combiners must be applied.

Let's assume we possess a second layer of combiners, $layer_2$. Naturally, $layer_2$ combiners' goal is to predict to which class the input instant belongs. Each combiner is practiced on a specific class and its task is to predict whether a given input instance belongs to its practiced class. For instance, $combiner_4$ is practiced on predicting whether an input instance belongs (or does not belong) to $class_4$. We choose to call the combiners of this layer "meta-classifiers".

For a successful fulfillment of $layer_2$'s task, it is required that its meta-classifiers will be able to distinguish between precisely three possible patterns of inputs. The first pattern is the 'general agreement' in which most of meta-classifier's inputs (the specialists' outputs predictions) agree; the second is the 'non-consent' in which meta-classifier inputs clearly do not agree on the instance's class. Literally, the meta-classifier inputs divide almost equally between those which are in favor of predicting

[8] Not including when base classifiers were trained in a one-against-one binarization

meta-classifier's practiced class and those which are not. The third pattern is the 'in-between' in which there is a clear majority in favor of one of two possible classes, but this majority can still not be considered as significant.

The inputs of each meta-classifier always form exactly one of the three possible patterns described above. When input values form a 'general agreement' pattern it is very easy for the meta-classifier to make a correct prediction. The correct prediction will be '*positive,*' meaning that the meta-classifier predicts that the Troika`s input instance belongs to the class on which it is practiced[9]. When input values form a 'non-consent' pattern it is also an easy task for the meta-classifier to make a correct prediction. The correct prediction will be '*negative,*' meaning that the meta-classifier predicts that the Troika`s input instance does not belong to the class on which it is practiced. When input values form an 'in-between' pattern it is not a simple task for the meta-classifier to make a correct prediction. The correct prediction may be '*positive*' or '*negative.*' A meta-classifier can be best evaluated by examining its accuracy rate trying to classify meta-classifier instances which usually forms this kind of pattern.

One might think that combining $layer_1$ classifiers using the voting algorithm would yield good results; however, this is not so. While voting may distinguish well between the 'general agreement' pattern and the 'non-consent' pattern, it is the 'in-between' pattern that will make the most prediction errors and unfortunately, it is this pattern that occurs most frequently when dealing with multi-class problems. A more complex model should then be induced. Its job will be to distinguish successfully between all three patterns. Once it has identified the 'in-between' pattern it has to make a correct prediction, sometimes against the majority of its inputs.

In Troika, we had chosen that meta-classifiers will use the one-against-all binarization method. We had used this method although it produces biased predictions. In fact, we actually needed this property and welcomed it. It ensures a low false positive rate in the meta-classifiers' predictions, because they will always be biased to predict '*negative.*' Optimally, we would like to have only one meta-

[9] Each Meta-classifier is practiced on one particular class and no two Meta classifiers are practiced on the same class.

classifier predict '*positive*' (and of course be correct) while all other Meta-classifiers predict '*negative*.' A more likely outcome is that more than one Meta-classifier will predict '*positive*' than that none will. Therefore, a fine bias may help push Troika's meta-classifiers to predict correctly.

Although being slightly biased, ultimately, it is possible that more than one meta-classifier will predict '*positive*' and it is also possible that none will, In that case, we cannot use $layer_2$ to produce a final decision regarding to which class an input instance belongs. This leads us to the conclusion that yet another layer of classifiers is needed. $Layer_3$ it is.

$Layer_3$'s task is to supply a final decision regarding to which class a Troika input instance belongs. It consists of one classifier; we call it the 'super classifier.' Its inputs are the meta-classifiers outputs. To fulfill its task it should learn about the meta-classifiers performance as a function of input instances.

In the next example we will assume a problem domain containing seven classes. The number of specialist classifiers therefore equals twenty one[10] and each meta-classifier will combine exactly six specialists[11]

In Figure 2, we show, without loss of generality, a schematic for a meta-classifier for class C_3. A specialist instance feeds to all specialists. Each specialist output is a prediction. The predictions are labeled $prediction_1$ to $prediction_6$. All six predictions are fed to the meta-classifiers for class C_3. After some calculations, this meta classifier will output its own prediction regarding the question whether the input instance should be labeled as class C_3 or not. An example of a "total agreement" pattern will be that any five or six out of six specialists will predict '*positive*.' If, on the other hand, three out of six Specialists predict '*positive*,' this will form the "non-consent" pattern.

---

[10] Having seven classes in a problem domain, we will train some $\frac{7(7-1)}{2} = 21$ specialists

[11] Meta-classifiers will always combine k-1 specialist classifiers predictions, and since k=7, each meta-classifier will combine six specialist predictions.

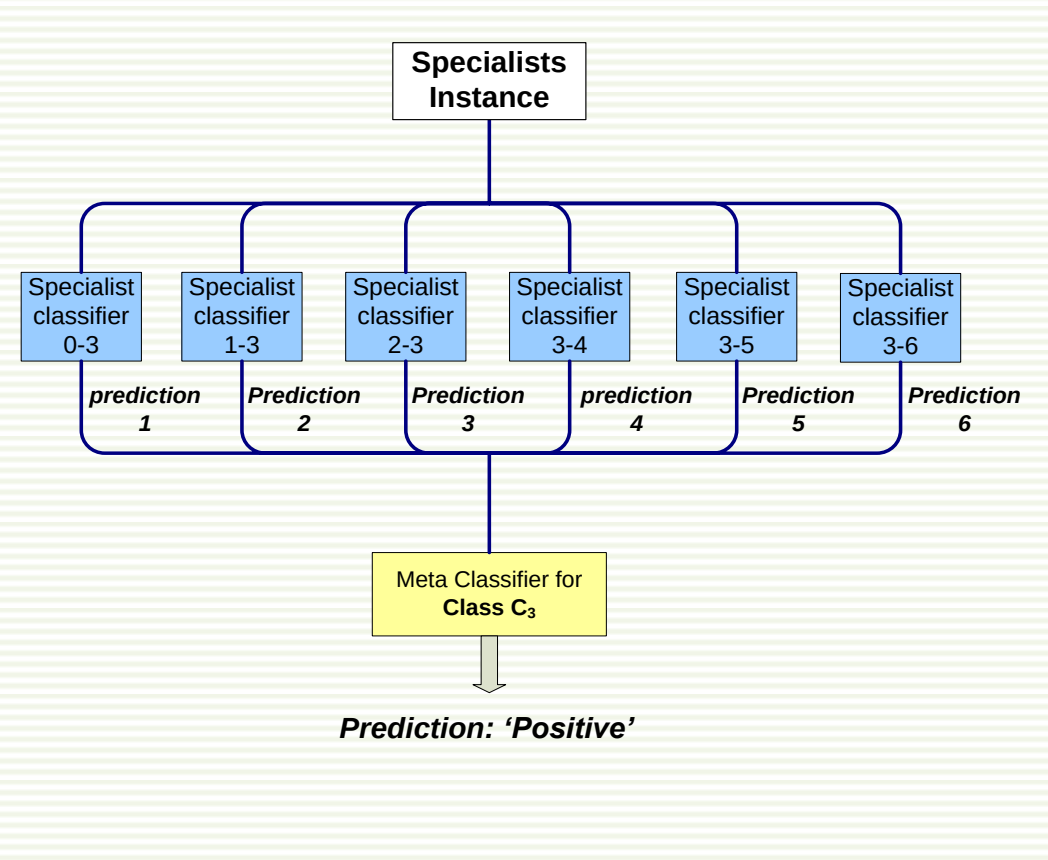


Figure 2: The schematic of Meta classifier for class $\mathbf{C_3}$

3.3. Troika's Training Process

Before one is able to use Troika to classify a new problem's instances, Troika, (and therefore all of its three combining layers of classifiers) must be trained. Algorithm 1 presents the proposed procedure. The most straightforward way of doing this is to train one layer of classifiers at a time. This leads to a training process which takes place in a sequence consisting of four stages. At any one stage, a different layer of classifiers will be trained. Since $layer_3$ depends on $layer_2$, $layer_2$ depends on $layer_1$ and $layer_1$ depend on $layer_0$, it is necessary first to train $layer_0$, then $layer_1$, followed by $layer_2$ and lastly $layer_3$.

Each layer is trained using a different dataset; first, $layer_0$ dataset is derived from the original dataset which was supplied to Troika as input; $layer_1$ dataset will be generated using predictions of $layer_0$ classifiers; $layer_2$ dataset will be generated using predictions of $level_1$ classifiers, and, finally, $level_3$ dataset will be generated

using predictions of $level_2$ classifiers. The technique of generating a derived dataset using predictions of classifiers will be discussed later.

Each layer is trained in a *k*-fold cross validation method. The process of training each layer (except for $layer_3$) is as follows: Start with splitting the layer's dataset into training-set and test-set. Then, all layer's classifiers are built using the training set (compound of *k*-1 parts of the layer's dataset). Next, the successor layer's dataset instances will be produced by applying the test-sets' instances on the layer's classifier. Later, all test-sets' instances are fed to layer's classifiers that will yield predictions. These predictions will be assembled to form a new instance for the successor dataset. A cross-validation fold is completed when the generating of successor instances from all the test-set's instances is finished. On the next fold, the new produced successor's dataset instances will be added to those of the previous fold. By the end of all *k* folds, the successor layer's dataset will contain exactly the same amount of instances as the present layer's dataset.

Finally, all layer's classifiers will be trained using the entire layer's datasets (not using only *k*-1 out of *k* parts of the dataset), so when Troika will be used for prediction, each layer will be trained on maximal number of instances.

```
1   Procedure: Train-Troika
2   Input: original-dataset: Dataset, folds: Integer
3   {
4       dataset←original-dataset
5       define Array-Of-Classifiers as array of classifiers
6       For layer = 0 to 3 do:
7       {
8           For each fold in folds do:
9           {
10              Split dataset to train-set and test-set  (according to current fold number)
11              If layer ≠3 Do:
12                  Array-Of-Classifiers←Train_SingleLayer( layer, train-set, folds)
13                  new-Instances←Classify(test-set, Array-Of-Classifiers)
14                  Add new-Instances to dataset[layer+1]
15              Else Train_SingleLayer( layer, train-set, folds)
16          }
17      layer = layer + 1
18      }
19  Train-Single-Layer(layer0, original-dataset) //Rebuild Base classifiers using
20                                                // the original dataset
21  }
```

Algorithm 1: Pseudo-Code of Troika Main Function

```
1   Procedure: Train-Single-Layer
2   Input: LayerNumber: Integer, dataset: Dataset, folds: Integer
3   Output: successor-Dataset: Dataset
4   {
5   Successor_Dataset ← emptyGroup
6   For each fold in folds do:
7     {
8          Split original-dataset to train-set and test-set
9         (according to current fold  number)
10        Build_Classifiers(LayerNumber , train-set)
11        For each instance in test-set
12        {
13             Produce probabilities-vector by applying instance on current layer's classifiers.
14             Generate a new Instance from probabilities-vector
15             Add the new Instance to successor-Dataset
16         }
17    }
18  Return successor-Dataset
19  }
```

Algorithm 2: Pseudo code of Troika training process for each one of its four layers, which is systematically called by the Main Procedure

3.4. Transformations of input Instance in Troika

There are four kinds of instance in Troika. The first type is the original training set. The second kind is the specialists' instances. These are derived from the base classifiers' predictions. The third kind of instance is the meta-classifiers' instances. These instances are derived from the specialists' predictions. The last kind of instance is the super classifier instance. These instances are derived from meta-classifiers predictions.

Given l is the number of base classifiers and k is the number of classes in the problem domain, each base classifier output vector, BC_l, can be presented as: $BC_l = \{P(C_0), P(C_0), ..., P(C_k)\}$. From these vectors, we produce the specialists' instances.

In general, specialist classifier, $Sp_{i\text{-}j}$, instances are composed using the probabilities P (C_i) and P (C_j) of each BC_i. It is possible also that one or more BC_l will not contain P (C_i) or P (C_j) or both. In this case, its prediction will not be included in the dataset for specialist classifier $Sp_{i\text{-}j}$.

$Classifier_1$		$Classifier_2$			$Classifier_l$		Class=*i*?
$P_{1,i,1}$	$P_{1,j,1}$	$P_{2,i,1}$	$P_{2,j,1}$	…	$P_{l,i,1}$	$P_{l,j,1}$	1
$P_{1,i,2}$	$P_{1,j,2}$	$P_{2,i,2}$	$P_{2,j,2}$	…	$P_{l,i,2}$	$P_{l,j,2}$	0
$P_{1,i,3}$	$P_{1,j,3}$	$P_{2,i,3}$	$P_{2,j,3}$	…	$P_{l,i,3}$	$P_{l,j,3}$	1
$P_{1,i,4}$	$P_{1,j,4}$	$P_{2,i,4}$	$P_{2,j,4}$	…	$P_{l,i,4}$	$P_{l,j,4}$	0
…	…	…	…	…	…	…	…
$P_{1,i,n}$	$P_{1,j,n}$	$P_{2,i,n}$	$P_{2,j,n}$	…	$P_{l,i,n}$	$P_{l,j,n}$	0

Table 5: Troika's dataset for Sp_{i-j} classifier

In Stacking, each example *m* (instance number *m*) of a Stacking meta-level dataset contains all, $P_{l,j,m}$[12], produced by *l* base classifier's for all *k* classes in the problem domain; there are therefore $k*l*n$ fields to the meta-classifier dataset. On the other hand, in Troika each instance *m* of Sp_{i-j} dataset contains only two values per base classifier, $P(C_i)$ and $P(C_j)$, and the average number of instances is $n' = \frac{2 \times n}{k}$ because, on average, $\frac{n}{k}$ instances are related with each class and each specialist's dataset contain instances related to two classes. Therefore, there are $2*l*\frac{2 \times n}{k}$ fields to a specialist dataset.

The reduction of dimensionality in Troika's first combining layer's dataset compared to the Stacking meta-level dataset is:

$$r = \frac{Specialist\_dataset\_volume}{stacking\_meta\_level\_dataset - volume} = \frac{4*l*\frac{n}{k}}{k*l*n} = \frac{4}{k^2} \qquad (2)$$

We can see that as *k*, the number of classes in the problem, increases there is a linear to *k* growth in Stacking's meta-level dataset while in Troika there is a linear to *k* decrease in specialists' dataset size. This is a big advantage for Troika, because it makes it possible to combine a very large group of base classifiers without being caught in the dimensionality course. In our experiments we used Troika and Stacking to combine as many as 3900 base classifiers (with letter dataset); Troika showed a

[12] $P_{l,j,m}$ refers to the probability given by base classifier *l* for class *j* on example number *m*

clear advantage in terms of accuracy. We suspect that the reason for Troika's triumph in this case derives from the huge dimensions (dimensionality course) of the meta-level dataset, making it hard for the Stacking meta-classifier to produce a good model. This weakness of Stacking had been shown in several previous studies and again in this study.

Given that each pair of classes *i* and *j* have a dedicated $Sp_{i\text{-}j}$, and *k* is the number of classes in the problem domain, there are some $\binom{k}{2}$ specialist classifiers in Troika. Each specialist classifier outputs a single prediction, $P_{i\text{-}j}$ *(inst)*, which stands for the probability, which was computed by $Sp_{i\text{-}j}$, that a specialist instance, *inst*, is of class *i*.

Troika has exactly *k* meta-classifiers, where *k* denotes the number of classes in the problem domain. Each meta-classifier has a different dataset that derives from a different projection of the output predictions of the specialists. Each meta dataset has one instance for each instance in the dataset which was given as an input to Troika. The meta-classifiers are trained using one-against-all binarization; for each meta instance, if the corresponding instance in the input dataset is of class C_k, then its class attribute is *positive*. Otherwise, the meta instance class attribute is *negative*. The attributes of each meta-classifier (meta-classifier$_k$ in general) instances are the probabilities $P_{i\text{-}j}$ *(inst)* produced by all specialist, $Sp_{i\text{-}j}$, where which *j* equals *k;* there are therefore always *k* attributes for each meta-classifier instance (not including the class attribute).

$Sp_{0\text{-}a}$	$Sp_{1\text{-}a}$		$Sp_{(a-1)\text{-}a}$	$Sp_{a\text{-}(a+1)}$		$Sp_{a\text{-}k}$	Class=*a*?
$P_{0,a,1}$	$P_{2,a,1}$	…	$P_{a\text{-}1,a,1}$	$P_{a,a+1,1}$	…	$P_{a,k,1}$	1
$P_{0,a,2}$	$P_{2,a,2}$	…	$P_{a\text{-}1,a,2}$	$P_{a,a+1,2}$	…	$P_{a,k,2}$	0
$P_{0,a,3}$	$P_{2,a,3}$	…	$P_{a\text{-}1,a,3}$	$P_{a,a+1,3}$	…	$P_{a,k,3}$	1
$P_{0,a,4}$	$P_{2,a,4}$	…	$P_{a\text{-}1,a,4}$	$P_{a,a+1,4}$	…	$P_{a,k,4}$	0
…	…	…	…	…	…	…	…
$P_{0,a,n}$	$P_{2,a,n}$	…	$P_{a\text{-}1,a,n}$	$P_{a,a+1,n}$	…	$P_{a,k,n}$	0

Table 6: Troika's Meta dataset for class$_a$

The volume of each meta dataset can be computed as follows:

$$V_{meta-dataset} = (k+1) * n \qquad (3)$$

where *k* is the number of classes in the problem's domain and *n* is the number of instances in the original dataset.

Comparing Troika's meta-datasets to a Stacking dataset is a rather tricky business, and depends on two major factors: the number of classes in the problem's domain and the number of base classifiers. The Stacking meta dataset is a function of the number of base classifiers. On the other hand, Troika's meta dataset is a function of the number of classes in the domain's problem. Our experience with the tested UCI datasets, shows that Troika's meta datasets tend to be much smaller than Stacking meta datasets.

StackingC's dataset volume is a function of the number of base classifiers. Each base classifier contributes one attribute to a meta instance; therefore, when an ensemble contains a large number of base classifiers (more than a few thousand), even though a much smaller one than Stacking meta dataset, StackingC 's dataset can grow to such an enormous size that it can no longer be used for training the meta-classifier. Troika, on the other hand, is much less sensitive to the number of base classifiers because each specialist is trained using the one-against-one binarization method. Conversely, Troika is more sensitive than StackingC to the number of classes in a domain's problem, in terms of training time. This is due to the amount of specialists that need to be trained: $\frac{k(k-1)}{2}$ (which yields time complexity of $O(k)^2$ for first combining layer) versus *k* meta-classifiers in StackingC (which yields time complexity of $O(k)$).

Given a meta-instance, each Troika meta-classifier$_j$[13] outputs a prediction, $P_j(inst)$, which reflects the belonging of the meta-instance, *inst*, to class C_j (therefore also the belonging of the original instance to that same class). It might be thought that each meta-classifier is responsible for the prediction for a single class; meta-classifier$_0$ is responsible for predicting the belonging of the input original instance to class$_0$. Meta-classifier$_1$ is responsible for predicting the belonging of the input original instance to class$_1$ etc.

[13] $j = \{1...k\}$

A vector of all meta-classifier predictions forms the super-instance:

$$\text{SuperInstance} = \{p_0(inst), p_1(inst), ..., p_k(inst), Class)\} \qquad (4)$$

Each instance in the super-dataset has a corresponding instance in the original dataset. The class attribute of the super dataset is copied from the corresponding instance of the original dataset without any changes. Table 7 shows the super dataset structure.

Meta-classifier$_0$	Meta-classifier$_1$		Meta-classifier$_k$	Class
$P_0(1)$	$P_1(1)$	…	$P_k(1)$	C_a
$P_0(2)$	$P_1(2)$	…	$P_k(2)$	C_b
$P_0(3)$	$P_1(3)$	…	$P_k(3)$	C_a
$P_0(4)$	$P_1(4)$	…	$P_k(4)$	C_c
…	…	…	…	…
$P_0(n)$	$P_1(n)$	…	$P_k(n)$	C_b

Table 7: Troika's Super dataset

4. Classifying using Troika

When a new instance, *x*, is input to Troika, firstly, it will be fed to all of the base-classifiers. Each base classifier will then process the given instance and produce their predictions, from which a specialist instance will be generated. $Sp_{i,j}inst$={$P_{i,j}$(x) | all base classifiers that were trained on classes *i* and *j*}. Next, each specialist, Sp_{i-j}, will classify its unique instance, $Sp_{i,j}inst$, (which derives directly from the base classifiers predictions) and produces a prediction $P_{i,j}(inst_{i-j})$; From these predictions *k* meta-instances, Meta-$inst_j$ (*j*=0..*k*) will be created; one for each of the meta-classifiers. Each meta-classifier$_i$, will then output its prediction, P$meta_i$(Meta-$inst_j$) and from these predictions will generate a super classifier instance, $\text{inst}_{\text{super}}$= {P$meta_0$($inst_j$) , P$meta_1$($inst_j$) ,…, P$meta_k$($inst_j$)} This single instance will be fed to the super classifier, which in return will produce its prediction, Troika's final prediction

$$FinalDecision(x) = \{P(C_1 \mid x), P(C_2 \mid x), ..., P(C_K \mid x)\} \qquad (5)$$

5. Evaluation Description

5.1. Experiment setup

In this chapter we will specify the conditions in which Troika had been tested. Our goal was to create a ground on which Troika could be correctly compared to Stacking and StackingC. We start with short overview on the datasets we used, the algorithms we implemented (Troika, Stacking and StackingC), how we trained each of them and what metrics we had used to measure the performance of the ensemble schemes. Next, we will display and review the results of the experiments in details.

5.1.1. Overview

The goal of our experiment was to measure the success of each one of the three ensemble methods (Troika, Stacking and StackingC) when applied to various problems.

The experiment had stretched into three dimensions. The first dimension was the number of inducers that were used to create the base classifiers upon which all the ensemble methods rely. The second was the different datasets, and the third dimension was the ensemble methods, of which we had three: Stacking, StackingC and Troika.

For the experiment to be valid, we had to split the training phase into two stages: first stage composed of base-classifiers training; in the second stage we had trained the ensembles, which got the base-classifiers from first stage as input. This means that all ensemble methods have been given the same base classifiers as input, therefore a valid comparison could successfully be made. This experiment could be given the title – which ensemble will combine the base-classifiers better?

5.1.2. Datasets

In total, we have used 29 different datasets in all the experiments; all of which were manually selected from the UCI repository (Merz and Murphy, 1998) and are widely used by the pattern recognition community for evaluating learning algorithms. The datasets vary across such dimensions as the number of target classes, of instances, of input features and their type (nominal, numeric).

Dataset	#Classes	#Instances	#Attributes
Anneal	6	898	39
Autos	5	202	26
Balance-scale	3	625	5
Diabetes	2	767	9
Flag	6	193	30
Glass[14]	5	205	10
heart-statlog	2	270	14
Hepatitis Domain	2	155	20
Votes	2	435	17
Ionosphere	2	351	35
Iris	3	150	5
KRKOPT	17	7,015	6
KR-vs-KP	2	3196	37
LED7	10	3,200	8
Letter[15]	26	2,500	17
Segment	7	2,310	20
Sonar	2	208	61
Soybean	19	683	36
Splice	3	3190	62
Vehicle	4	846	19
Vowel	11	990	14
Waveform	3	5,000	41
Zoo	7	101	18

Table 8: Datasets used for evaluation

5.1.3. Ensemble Algorithms Examined

The examined ensemble schemes - Troika, Stacking and StackingC were implemented in WEKA (Witten and Frank, 2005) in JAVA programming language. The implementation of the Troika algorithm can be downloaded from: http://www.ise.bgu.ac.il/faculty/liorr/troika/.

5.1.4. Combining classifiers

All classifiers in Troika, Stacking and StackingC which participate in combining the base-classifiers, i.e., the Specialist classifiers, Meta classifiers and the Super classifier (in Troika) and meta classifiers (in Stacking and StackingC) where

[14] 'Glass,' was adjusted to have fewer classes because some of their original classes had too few instances (less than 10), which yields a very poor result.

[15] The 'Letter' dataset was too big to fit into memory, so we reduced its volume using a sample a smaller dataset which we could use to investigate all ensemble methods.

induced using Logistic algorithm. We have chosen this particular algorithm after trying various alternative inducers.

5.1.5. Base classifiers training process

First, we used multiple inducers of different branches of the machine-learning theory. Generally, we intended to use only six inducers, C4.5 (trees), SMO (function), IB1 (lazy), VFI (Misc.), BayesNet (Bayes), PART (Rules), but we tested also other configurations: three inducers (C4.5, IB1 and VFI) and one inducer (C4.5).

Second, All base classifiers were trained using the one-against-one binarization method in which, typically, $\frac{k(k-1)}{2}$ base classifiers are trained; that is one classifier for each pair of classes. Secondly, instead of training a solely single base-classifier for each pair of classes, we actually trained two[16], each with difference training instances. Train-set$_1$ (the train-set of first base-classifier, which derives from the train-set) contained the first 125% of the instances in a cyclic manner (that is, the next instance past the last is the first again), whereas train-set$_2$ contains the next 125% group of the instances in the same cyclic manner[17]. After creating the 125% instances training-set, the training process continued normally, i.e. using the *k*-fold cross-validation process. Using this process, we actually trained $k(k-1)$ base-classifiers for each inducer.

Inst#	Instance
1	Instance$_1$
2	Instance$_2$
.	.
.	.
.	.
.	.
100	Instance$_{100}$

Train-set

Inst#	Instance
1	Instance$_1$
2	Instance$_2$
.	.
.	.
.	.
.	.
100	Instance$_{100}$
101	Instance$_1$
102	Instance$_2$

Inst#	Instance
1	Instance$_{26}$
2	Instance$_{27}$
.	.
.	.
.	Instance$_{100}$
.	Instance$_1$
100	Instance$_2$
101	.
102	.

[16] We had trained two classifiers instead of a single one to increase the number of different classifiers that were combined. Diversity among base classifiers was found to be a contributing factor for the performance of classifiers combination [Zenobi et. el.].

17 First train-set$_2$ instance is the subsequent instance of the last instance of train-set$_1$

.	.
.	.
125	$Instance_{25}$

$Train\text{-}set_1$

.	.
.	.
125	$Instance_{50}$

$Train\text{-}set_2$

Table 9: depict the composition of train-set1 and train-set2 where given an arbitrary train-set, which contains 100 instances.

Later on, we suspected that it could be that our choice of training the base-classifiers using one-against-one binarization method might have been the prime reason why Troika preformed better than its rivals, so we repeated the experiment using one-against-all binarization and non-binarization methods. Later, in Tables 13, 14 and 15 we show those experiments results.

5.1.6. Metric measured

In this experiment the following metrics were measured:

- **Accuracy**: Accuracy is the rate of correct (incorrect) predictions made by a model over a data set. In order to estimate the generalized accuracy, a 10-fold cross-validation procedure[18] was repeated 5 times. For each 10-fold cross-validation, the training set was randomly partitioned into 10 disjoint instance subsets. Each subset was utilized once in a test set and nine times in a training set. The same cross-validation folds were implemented for all algorithms. Since the mean accuracy is a random variable, the confidence interval was estimated by using the normal approximation of the binomial distribution. Furthermore, the one-tailed paired t-test with a confidence level of 95% verified whether the differences in accuracy between the Troika algorithm and the other algorithms were statistically significant. In order to conclude which algorithm performs best over multiple datasets, we followed the procedure proposed in Demsar (2006). In the case of multiple classifiers we first used the adjusted Friedman test in order to reject the null hypothesis and then the Bonferroni–Dunn test to examine whether the new algorithm performs significantly better than existing algorithms.

[18] The 10-folds cross validations refers to the process in which 10 different Troika models where trained. The k-folds cross validation process mentioned before is the inner process of troika training. When the 10-folds cross validation process (for assessing the performance of Troika) is finished, Troika is trained once again, using all instances in the dataset.

- **Area under ROC curve**: The second measure we used to evaluate Troika is the Area under the ROC (Receiver Operating Characteristic) curve, a graphical plot of the sensitivity vs. (1 - specificity) for a binary classifier system as its discrimination threshold is varied. The ROC can also be represented equivalently by plotting the fraction of true positives (TPR = true positive rate) vs. the fraction of false positives (FPR = false positive rate). ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution. ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making. Widely used in medicine, radiology, psychology and other areas for many decades, it has been introduced relatively recently in other areas such as machine learning and data mining.
- **Training time**: This measure is an applicative one. It has two significances; first, and most logical, heavy time consumption is bad. We would prefer a fast learning ensemble that will yield the best accuracy or area under ROC. Second, the longer time the training of an ensemble takes, the more CPU time it requires, and thus, the more energy it consumes. This is very important on mobile platforms that may be using an ensemble for various reasons

5.1.7. Ensemble Size

Since the accuracy and the classifier complexity are affected by the ensemble size (number of classifiers), we examined three ensemble configurations: six, three, and one inducers. The size of the whole ensemble, n_{en}, can be described in the next equation:

$$n_{en} = n_{ind} * k(k-1) \qquad (7)$$

where n_{ind} is the number of inducers and *k* is the number of classes in dataset. For example, the size of Troika ensembles on 'letter' dataset, which contains 26 classes is $n_{en} = 6 * 26(26-1) = 3900$ when six inducers are been used, 1950 and 975 inducer for three and one inducers respectively.

5.2. Results

Tables 10, 11 and 12 present the results obtained using different number of inducers using the 10-fold cross-validation procedure which was repeated five times and one-against-one binarization method. The superscript "+" indicates that the degree of accuracy of Troika was significantly higher than the corresponding algorithm at a confidence level of 95%. The "–" superscript indicates the accuracy was significantly lower. In Table 10 we used six inducers (C4.5, PART, VFI, IBk, Bayes-Net and SMO) for creating base classifiers, in Table 11 we used three inducers (C4.5, VFI, IBk) and finally in Table 12 we used only one inducer (C4.5). The reason why we repeat all the testing three times with different amount of inducers is to investigate the effect of inducer number on ensemble performance. In Tables 10-12 “Best B.C.” stands for best base classifier chosen by cross validation.

Table 10: Comparing ensemble algorithms using 6 inducers: C4.5 (Trees), PART (rules), VFI (misc.), IBk (Lazy), Bayes-Net (Bayes), SMO (functions).

Dataset	Accuracy				Area Under ROC			Execution time		
	Stacking	StackingC	Troika	Best B.C.	Stacking	StackingC	Troika	Stacking	StackingC	Troika
Heart-statlog	83.80 ± 6.22	83.93 ± 6.10	84 ± 6.25	83.89	0.90 ± 0.06	0.90 ± 0.06	0.90 ± 0.06	1.70 ± 0.42	1.65 ± 0.06	1.74 ± 0.08
Diabetes	75.91 ± 3.91	75.91 ± 3.91	75.85 ± 3.97	77.1	0.82 ± 0.04	0.82 ± 0.04	0.82 ± 0.04	-3.22 ± 0.11	4.08 ± 0.09	4.19 ± 0.12
Sonar	78.74 ± 11.01	78.74 ± 11.01	85.05 ± 10.85	86.17	0.85 ± 0.11	0.85 ± 0.11	0.85 ± 0.11	1.10 ± 0.95	1.42 ± 1.27	2.32 ± 2.17
Hepatitis	69.22 ± 10.78	69.22 ± 10.78	69.6 ± 11.02	69.71	0.73 ± 0.13	0.73 ± 0.13	0.73 ± 0.13	1.70 ± 0.54	1.82 ± 0.50	2.14 ± 0.89
Votes	96.06 ± 3.14	96.06 ± 3.14	96.13 ± 2.95	96.57	0.99 ± 0.02	0.97 ± 0.03	0.99 ± 0.02	-1.72 ± 0.08	2.25 ± 0.08	2.29 ± 0.10
Ionosphere	93.25 ± 4.12	93.25 ± 4.12	93.14 ± 4.09	94.5	0.97 ± 0.03	-1.00 ± 0.00	0.97 ± 0.03	-2.20 ± 0.13	-2.84 ± 0.12	3.24 ± 0.25
Kr-vs-kp	99.51 ± 0.42	99.53 ± 0.37	99.48 ± 0.39	99.44	-1.00 ± 0.00	+0.82 ± 0.04	0.99 ± 0.00	-62.54 ± 4.18	+112.03 ± 1.64	90.99 ± 2.39
balance-scale	93.76 ± 2.84	90.63 ± 2.37	93.87 ± 2.84	90.53	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	6.90 ± 0.18	7.42 ± 0.19	7.74 ± 19.14
Iris	94.13 ± 6.01	87.33 ± 9.06	95.2 ± 5.53	96.27	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.01	-1.29 ± 0.11	1.64 ± 0.16	1.74 ± 0.28
splice	+90.87 ± 2.04	+84.95 ± 3.02	96.1 ± 1.04	95.54	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	-187.6 ± 39.1	+714.14 ± 10.69	582.31 ± 11.11
Waveform	84.64 ± 1.86	82.88 ± 1.26	85.56 ± 2.2	80.01	0.95 ± 0.01	0.96 ± 0.01	0.95 ± 0.01	-503.0 ± 130.7	+1640.30 ± 4.09	684.38 ± 2.79
Vehicle	73.00 ± 4.72	69.56 ± 6.59	75.59 ± 5.77	74.08	0.79 ± 0.09	0.78 ± 0.11	0.85 ± 0.05	+425.9 ± 997.1	+42.52 ± 12.04	24.70 ± 7.48
Glass(5 classes)	+49.40 ± 12.64	+60.00 ± 10.56	71.15 ± 9.37	70.27	+0.75 ± 0.10	0.80 ± 0.10	0.91 ± 0.06	+30.8 ± 5.84	5.52 ± 0.66	6.15 ± 0.23
Autos	57.12 ± 11.32	60.98 ± 11.42	78.83 ± 10.82	81.77	0.89 ± 0.13	0.94 ± 0.06	0.96 ± 0.08	+43.2 ± 8.2	+17.96 ± 1.55	10.04 ± 0.28
Flag	+38.60 ± 10.82	46.84 ± 12.16	62.46 ± 7.41	60.72	0.64 ± 0.14	0.72 ± 0.11	0.79 ± 0.12	+207.7 ± 32.7	+48.49 ± 11.55	17.02 ± 1.01
Anneal	97.25 ± 1.57	+89.82 ± 4.97	97.33 ± 1.84	95.49	0.79 ± 0.30	0.92 ± 0.19	0.82 ± 0.26	-23.1 ± 0.2	+97.02 ± 1.47	68.21 ± 1.75
Zoo	99.30 ± 2.68	98.56 ± 3.75	98.93 ± 3.28	98.9	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.01	+21.2 ± 5.9	5.63 ± 0.12	4.45 ± 0.21
Segment	96.89 ± 0.90	+87.67 ± 4.67	97.71 ± 0.83	96.79	0.99 ± 0.01	0.99 ± 0.01	1.00 ± 0.01	+211.7 ± 12.3	+1327.59 ± 63.83	163.26 ± 10.59
LED7	71.53 ± 3.04	+57.99 ± 5.90	71.17 ± 3.58	73.59	0.91 ± 0.05	0.89 ± 0.07	0.93 ± 0.06	+353.1 ± 14.4	+602.48 ± 22.99	299.25 ± 1.51
vowel	+76.69 ± 4.84	+66.85 ± 5.88	94.43 ± 2.23	99.05	1.00 ± 0.01	1.00 ± 0.00	0.99 ± 0.03	61.2 ± 3.8	91.81 ± 3.56	464.64 ± 286.28
KRKOPT (25%)	+50.74 ± 2.03	+34.53 ± 1.84	56.99 ± 1.91	59.11	0.74 ± 0.04	0.77 ± 0.07	0.82 ± 0.04	+10196 ± 63.9	1356.86 ± 42.26	7391.03 ± 34.06
Soybean	90.87 ± 3.80	+74.62 ± 5.22	89.51 ± 4.18	93.1	+1.00 ± 0.00	+1.00 ± 0.00	1.00 ± 0.00	1244.2 ± 269.4	2033.01 ± 920.61	1003.86 ± 188.51
Letter (12.5%)	66.94 ± 2.86	+40.22 ± 2.75	71.24 ± 3.14	85.2	0.94 ± 0.04	0.93 ± 0.09	0.94 ± 0.09	-450.0 ± 17.7	+3768.95 ± 122.76	1517.95 ± 44.92
Average UCI	79.49 ± 4.96	75.22 ± 5.69	**84.02 ± 4.59**	82.360	0.90 ± 0.06	0.91 ± 0.05	**0.92 ± 0.05**	610.4 ± 69.9	516.85 ± 53.14	537.11 ± 26.79

Table 11: Comparing ensemble algorithms using 3 inducers: C4.5 (Trees), VFI (misc.), IBk (Lazy)

Dataset	Accuracy				Area Under ROC			Execution time		
	Stacking	StackingC	Troika	Best B.C.	Stacking	StackingC	Troika	Stacking	StackingC	Troika
heart-statlog	82.59 ± 6.44	82.59 ± 6.44	82.52 ± 6.58	79.89	0.90 ± 0.06	0.90 ± 0.06	0.90 ± 0.06	-0.84 ± 0.07	0.97 ± 0.05	0.99 ± 0.05
Diabetes	75.95 ± 4.21	75.95 ± 4.21	75.98 ± 4.06	74.92	0.82 ± 0.05	0.82 ± 0.05	0.82 ± 0.05	-2.08 ± 0.07	+2.70 ± 0.07	2.56 ± 0.07
Sonar	85.73 ± 8.67	85.73 ± 8.67	85.68 ± 8.76	86.17	0.92 ± 0.07	0.92 ± 0.07	0.92 ± 0.07	-1.55 ± 0.07	+1.94 ± 0.07	1.80 ± 0.06
Hepatitis	69.09 ± 12.38	69.09 ± 12.38	69.16 ± 12.66	67.98	0.75 ± 0.14	0.75 ± 0.14	0.75 ± 0.14	0.72 ± 0.05	0.72 ± 0.05	0.76 ± 0.04
Votes	95.58 ± 2.94	95.58 ± 2.94	95.58 ± 2.96	96.57	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	-1.23 ± 0.06	1.60 ± 0.06	1.54 ± 0.05
Ionosphere	91.17 ± 4.45	91.17 ± 4.45	91.22 ± 4.46	94.5	0.95 ± 0.04	0.95 ± 0.04	0.95 ± 0.04	-1.66 ± 0.07	+2.17 ± 0.06	2.02 ± 0.07
Kr-vs-kp	99.45 ± 0.46	99.52 ± 0.33	99.45 ± 0.44	99.44	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	-66.47 ± 4.70	+124.56 ± 5.35	89.64 ± 4.28
balance-scale	92.45 ± 2.87	89.14 ± 2.16	91.04 ± 3.00	86.72	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	+4.75 ± 0.39	+5.34 ± 0.52	3.07 ± 0.13
Iris	93.60 ± 5.68	88.00 ± 9.48	95.47 ± 4.63	96.07	1.00 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	+1.65 ± 0.27	+0.97 ± 0.06	0.79 ± 0.05
Splice	93.98 ± 1.64	+85.96 ± 2.45	95.99 ± 0.92	94.03	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	-98.11 ± 6.20	+620.25 ± 5.70	489.20 ± 2.43
Waveform	83.77 ± 1.61	+79.90 ± +1.78	84.45 ± 1.70	77.67	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	-97.54 ± 0.83	+168.74 ± 0.96	116.21 ± 1.50
Vehicle	72.05 ± 4.77	71.26 ± 5.23	75.12 ± 3.57	72.28	0.85 ± 0.05	0.84 ± 0.04	0.86 ± 0.03	+46.42 ± 17.21	+23.74 ± 5.85	10.86 ± 0.15
Glass(5 classes)	+53.88 ± 14.59	68.29 ± 10.76	74.22 ± 9.50	70.27	0.84 ± 0.11	0.90 ± 0.07	0.92 ± 0.07	+419.9 ± 708.8	4.57 ± 1.42	3.11 ± 0.31
Autos	+59.80 ± 12.31	+62.74 ± 10.21	79.39 ± 8.92	81.77	0.90 ± 0.12	0.93 ± 0.10	0.94 ± 0.09	+27.83 ± 55.89	+9.40 ± 0.51	6.26 ± 0.52
Flag	+40.68 ± 12.33	50.58 ± 12.24	61.00 ± 9.44	57.32	0.63 ± 0.14	0.75 ± 0.11	0.78 ± 0.10	+94.78 ± 26.44	+17.61 ± 1.19	8.37 ± 0.12
Anneal	97.81 ± 2.00	+92.50 ± 3.02	97.96 ± 1.47	95.49	0.85 ± 0.29	0.88 ± 0.25	0.95 ± 0.13	-12.62 ± 0.82	+60.54 ± 0.31	39.68 ± 0.45
Zoo	97.94 ± 4.72	98.57 ± 3.73	96.28 ± 7.01	98.9	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.04	+14.32 ± 1.88	+3.35 ± 0.07	2.53 ± 0.07
Segment	96.93 ± 1.18	+83.10 ± 4.07	97.61 ± 0.91	96.79	0.99 ± 0.01	0.99 ± 0.01	1.00 ± 0.01	+187.66 ± 11	+1184.7 ± 198.1	136.76 ± 2.63
LED7	71.63 ± 2.43	+56.72 ± 6.01	71.51 ± 2.72	73.48	0.91 ± 0.05	0.88 ± 0.05	0.93 ± 0.04	+265.29 ± 3.07	+458.19 ± 3.03	224.50 ± 2.39
Vowel	+63.98 ± 4.99	+41.43 ± 4.90	74.46 ± 5.05	99.05	0.93 ± 0.07	0.94 ± 0.08	0.97 ± 0.04	+393.23 ± 1.72	+726.97 ± 3.12	300.10 ± 1.47
KRKOPT (25%)	+48.49 ± 7.03	+32.51 ± 3.29	65.33 ± 8.32	59.11	-0.69 ± 0.04	0.79 ± 0.09	0.82 ± 0.06	-596.32 ± 6.19	+3432.22 ± 6.08	1667.22 ± 3.19
Soybean	90.68 ± 3.10	+67.84 ± 4.67	87.99 ± 3.60	93.06	0.99 ± 0.05	1.00 ± 0.00	0.98 ± 0.06	+1723.3 ± 69.8	+3075.1 ± 203.2	1111.73 ± 35.3
Letter (12.5%)	+63.29 ± 3.49	+34.17 ± 3.57	77.65 ± 3.04	85.2	0.94 ± 0.06	0.94 ± 0.08	0.94 ± 0.05	-434.49 ± 18.47	+3569.6 ± 233.3	1243.56 ± 39.66
Average UCI	80.32 ± 5.19	74.91 ± 5.41	84.19 ± 4.86	82.360	0.91 ± 0.06	0.92 ± 0.06	0.93 ± 0.05	187.25 ± 38.91	586.71 ± 29.09	223.51 ± 14.94

Table 12: Comparing ensemble algorithms using one inducer, C4.5, summary of experimental results for UCI repository datasets.

Dataset	Accuracy				Area Under ROC			Execution time		
	Stacking	StackingC	Troika	Best B.C.	Stacking	StackingC	Troika	Stacking	StackingC	Troika
Heart-statlog	76.33 ± 7.41	76.33 ± 7.41	76.33 ± 7.41	78.15	0.81 ± 0.09	0.81 ± 0.09	0.81 ± 0.09	-0.36 ± 0.06	-0.39 ± 0.06	0.52 ± 0.07
Diabetes	71.44 ± 3.89	71.44 ± 3.89	71.52 ± 3.94	74.92	0.74 ± 0.06	0.74 ± 0.06	0.74 ± 0.06	-0.73 ± 0.07	0.86 ± 0.14	1.01 ± 0.10
Sonar	72.08 ± 9.03	72.08 ± 9.03	72.08 ± 9.03	73.61	0.77 ± 0.10	0.77 ± 0.10	0.77 ± 0.10	-0.34 ± 0.05	0.37 ± 0.06	0.51 ± 0.10
Hepatitis	61.86 ± 12.23	61.86 ± 12.23	61.86 ± 12.23	63.18	0.63 ± 0.15	0.63 ± 0.15	0.63 ± 0.15	0.32 ± 0.05	0.34 ± 0.06	0.44 ± 0.08
Votes	95.63 ± 2.82	95.63 ± 2.82	95.63 ± 2.82	96.57	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	-0.34 ± 0.06	-0.40 ± 0.06	0.57 ± 0.09
Ionosphere	89.20 ± 4.46	89.20 ± 4.46	89.20 ± 4.46	89.74	0.91 ± 0.05	0.91 ± 0.05	0.91 ± 0.05	-0.35 ± 0.05	-0.39 ± 0.07	0.53 ± 0.07
Kr-vs-kp	99.55 ± 0.34	99.56 ± 0.31	99.61 ± 0.32	99.44	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	-5.66 ± 0.26	-10.01 ± 0.31	11.79 ± 0.62
Balance-scale	81.08 ± 3.45	79.40 ± 3.37	81.08 ± 3.45	77.82	0.94 ± 0.04	0.86 ± 0.05	0.93 ± 0.04	1.28 ± 0.15	1.18 ± 0.16	1.40 ± 0.11
Iris	93.60 ± 5.52	+66.00 ± 2.01	93.60 ± 5.52	94.73	0.99 ± 0.02	0.99 ± 0.03	0.99 ± 0.02	-0.35 ± 0.08	-0.33 ± 0.07	0.58 ± 0.12
Splice	94.08 ± 1.53	93.59 ± 1.84	94.26 ± 1.70	94.03	0.96 ± 0.02	0.97 ± 0.01	0.97 ± 0.01	+195.33 ± 1.67	-9.14 ± 1.59	13.46 ± 1.60
Waveform	75.62 ± 2.34	+68.30 ± 2.16	75.56 ± 2.34	75.08	+0.99 ± 0.02	+0.98 ± 0.03	0.86 ± 0.02	-42.25 ± 0.52	+81.18 ± 0.76	53.73 ± 0.57
Vehicle	74.68 ± 4.56	70.63 ± 4.63	74.68 ± 4.56	72.28	0.86 ± 0.04	0.84 ± 0.05	0.84 ± 0.04	+3.20 ± 0.10	-2.11 ± 0.12	2.58 ± 0.14
Glass(5 classes)	66.42 ± 11.07	64.64 ± 9.79	70.19 ± 9.43	67.63	0.86 ± 0.08	0.87 ± 0.07	0.87 ± 0.08	3.91 ± 13.40	0.58 ± 0.28	0.84 ± 0.19
Autos	72.15 ± 9.67	55.81 ± 9.16	72.15 ± 9.67	81.77	0.95 ± 0.07	0.96 ± 0.06	0.95 ± 0.07	+2.27 ± 0.18	1.18 ± 0.13	1.28 ± 0.14
Flag	60.74 ± 10.08	58.21 ± 7.80	60.74 ± 10.08	56.71	0.80 ± 0.10	0.83 ± 0.09	0.81 ± 0.10	+5.26 ± 1.62	1.67 ± 0.15	1.81 ± 0.16
Anneal	90.25 ± 9.90	83.07 ± 5.07	93.81 ± 6.10	92.35	0.92 ± 0.18	0.92 ± 0.20	0.93 ± 0.19	+17.31 ± 11.30	-0.77 ± 0.04	1.92 ± 0.26
Zoo	93.93 ± 7.21	+78.69 ± 6.69	93.93 ± 7.21	92.61	0.86 ± 0.02	0.86 ± 0.02	0.97 ± 0.10	+1.10 ± 0.22	-0.54 ± 0.06	0.73 ± 0.11
Segment	94.49 ± 2.79	+42.72 ± 5.14	95.84 ± 2.58	96.93	1.00 ± 0.00	1.00 ± 0.01	1.00 ± 0.01	-35.89 ± 0.20	-2.25 ± 0.13	67.79 ± 4.59
LED7	70.56 ± 2.84	+24.84 ± 5.13	70.98 ± 2.94	73.34	0.90 ± 0.04	0.91 ± 0.05	0.93 ± 0.03	+153.77 ± 0.86	-8.90 ± 0.17	38.72 ± 5.21
vowel	74.72 ± 4.61	+35.13 ± 3.49	79.82 ± 4.18	80.2	0.97 ± 0.03	0.97 ± 0.03	0.99 ± 0.01	-48.21 ± 2.96	-3.24 ± 0.77	287.01 ± 206.10
KRKOPT (25%)	49.15 ± 2.11	+17.50 ± 1.26	52.45 ± 2.61	54.69	0.73 ± 0.05	0.83 ± 0.07	0.83 ± 0.05	+1671.34 ± 7.81	+3283.85 ± 14.10	1374.35 ± 10.69
Soybean	90.66 ± 4.02	+22.73 ± 0.88	90.80 ± 6.93	91.78	0.99 ± 0.05	1.00 ± 0.01	1.00 ± 0.00	19.36 ± 1.18	20.51 ± 1.10	20.33 ± 2.18
Letter (12.5%)	58.56 ± 2.68	+7.37 ± 0.53	62.32 ± 5.67	73.19	0.88 ± 0.10	0.93 ± 0.07	0.91 ± 0.08	-193.57 ± 6.16	+342.59 ± 6.03	218.66 ± 2.13
Average UCI	79.79 ± 5.20	64.90 ± 4.58	80.29 ± 5.50	79.254	0.89 ± 0.06	0.90 ± 0.06	0.90 ± 0.06	96.72 ± 1.97	150.96 ± 1.06	84.12 ± 9.43

5.2.1. Results analysis - using six inducers

There are few datasets in which the Troika obtained a degree of accuracy lower to that of Stacking and StackingC, but none are significant. There are cases in which Troika achieved much higher accurate results compare to the other two ensemble methods (Sonar, Splice, Glass, Flag, Vowel, and KRKOPT).

A statistical analysis of the accuracy results on the entire dataset collection indicates that (1) in five datasets Stacking achieved significantly lower accuracy compare to Troika's; (2) in none of the datasets Stacking excels Troika; (3) in nine datasets StackingC achieved significantly lower accuracy compare to Troika's; (4) in none of the datasets StackingC excels Troika.

A statistical analysis of the Area under ROC curve results of the entire dataset collection indicates that (1) Stacking and StackingC achieved significantly lower results compare to Troika in two datasets; (2) Stacking and StackingC were better than Troika in one dataset.

Although mean execution time of Troika is longer than stackingC and shorter than of Stacking, adjusted non-parametric Friedman test with a confidence level of 95% shows that those differences are not significant.

The null-hypothesis that all ensemble methods and the best classifier perform the same using six inducers was rejected using the adjusted non-parametric Friedman test with a confidence level of 95%. Using the Bonferroni-Dunn test we could reject the null-hypothesis that Troika and Stacking perform the same at confidence levels of 93.4%. Using same test we could also reject the hypothesis that Troika and StackingC performs the same at confidence levels above 99%. Finally, the Bonferroni-Dunn test had shown that there was no significant difference between Troika and the best classifier.

5.2.2. Results analysis - using three inducers

Using Troika in conjunction with three inducers yielded results that resemble those shown in Table 10 where we used six inducers.

A statistical analysis of the accuracy results on the entire dataset collection indicates that (1) in six datasets Stacking achieved significantly lower accuracy compare to Troika's; (2) in none of the datasets Stacking excels Troika; (3) in ten datasets StackingC achieved significantly lower accuracy compare to Troika's; (4) in none of the datasets StackingC excels Troika.

Statistical analysis of the ROC results of the entire dataset collection indicates no significant difference between all the ensemble schemes, although Troika has a trifle advantage on average.

Statistical analysis of the execution time reveals that there is a difference between ensemble methods. Using the Bonferroni-Dunn test with a confidence level of 95% shows Troika execute time is shorter compares to StackingC.

The null-hypothesis that all ensemble methods perform the same using three inducers was rejected using the adjusted non-parametric Friedman test with a confidence level of 95%. Using the Bonferroni-Dunn test we could reject the null-hypothesis that Troika and Stacking perform the same at confidence levels of 92%. Using same test we could also reject the hypothesis that Troika and StackingC performs the same at confidence levels above 99%. The Bonferroni-Dunn test had shown that there was no significant difference between Troika and the best classifier.

5.2.3. Results analysis - using one inducer

Using one inducer yielded very different results compare to using three or six inducers.

Statistical analysis of the ROC results of the entire dataset collection indicates no significant difference between all the ensemble schemes and the best classifier.

Statistical analysis of the execution time reveals that there is a difference between the ensemble methods. Using the Bonferroni-Dunn test with a confidence level of 95% shows Troika has a longer execute time compares to Stacking and StackingC.

Figures 3, 4, 5 and 6 present the four Troika‘s ROC graphs, computed from the results on Vehicle dataset using 6 inducers. Each graph belongs to one of Vehicle’s class.

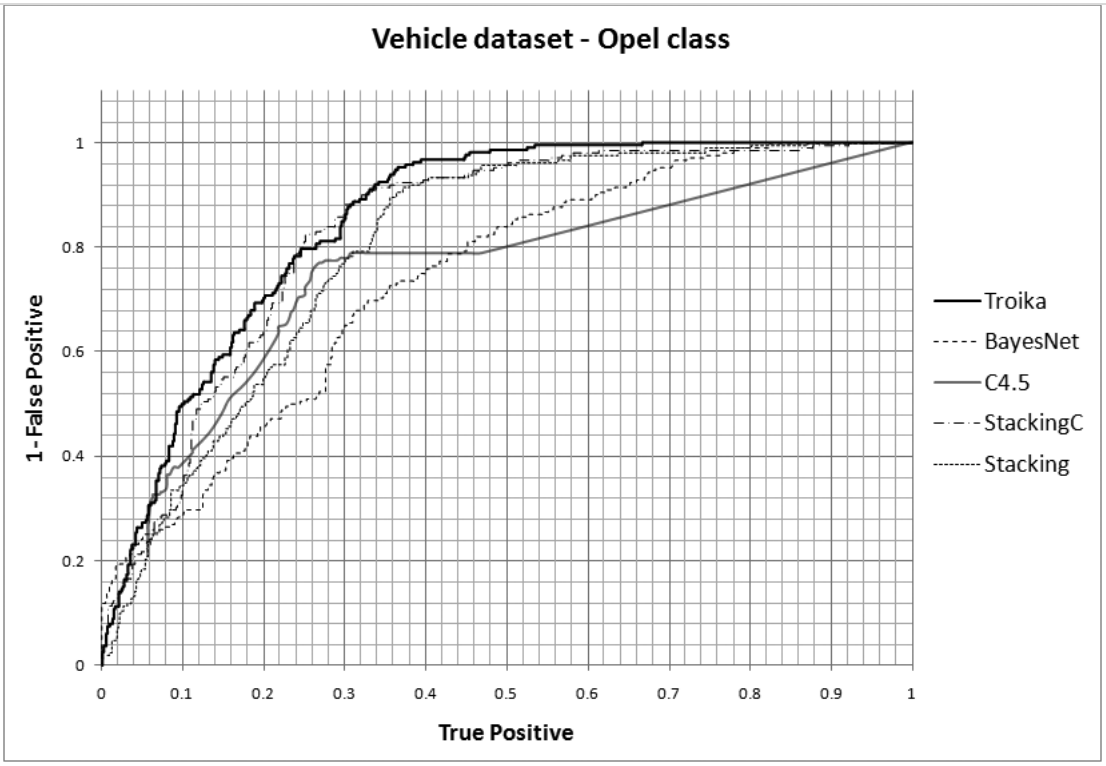


***Figure 3*:** ROC graph for “Opel” class. It can be seen that Troika, in the strong line, excels its opponents in almost every segments of the graph.

***Figure* 4:** ROC graph for "Bus" class.

***Figure 5*:** ROC graph for “Saab” class

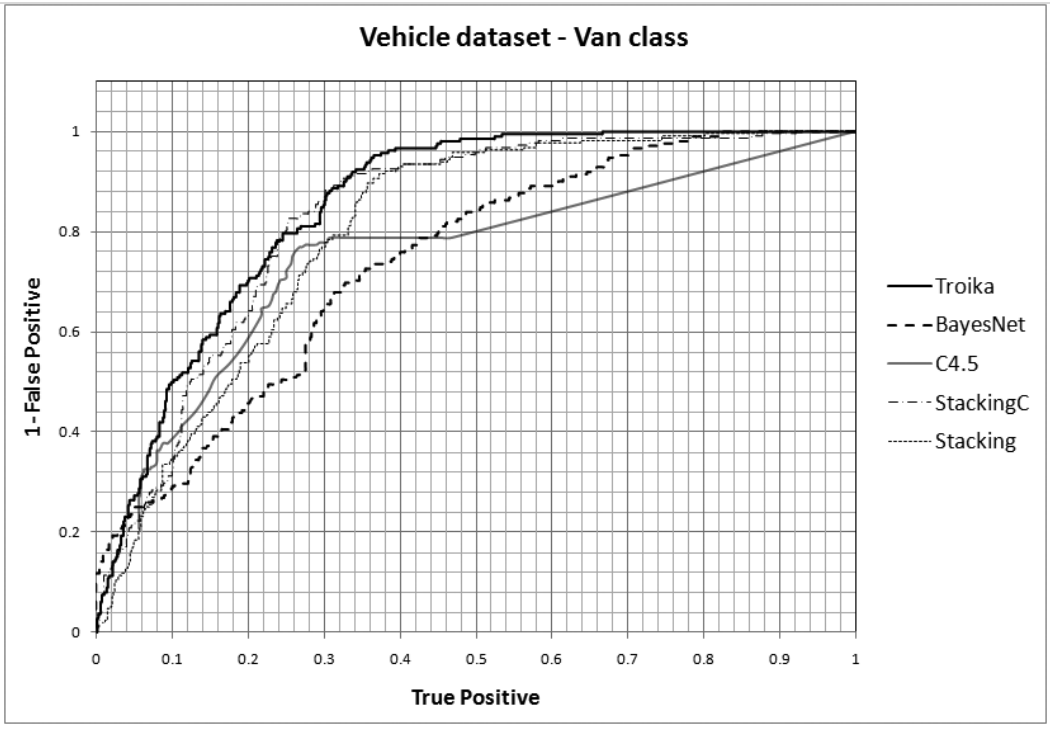


***Figure 6*:** ROC graph for “Van” class

We can observe from the graphs above that there is a consistent advantage to Troika in all classes of Vehicle dataset. Although, in general, we found no significant difference between all three tested ensemble schemes concerning area under ROC graph, there is an advantage to Troika in multiclass datasets where there is sufficient number of instances in dataset.

5.3. How base-classifiers class binarization affect ensemble accuracy?

As indicated in Section 2.6, there are several methods for converting multiclass classification tasks into binary classification tasks. There are two reasons why we had made the effort to experiment different kinds of class binarization methods. First, recall that our primary requirement from the base classifier was that they will be given as an input to all ensemble schemas uniformly, so that an acceptable comparison between all ensemble schemes could be made. This, we had successfully implemented by training the base classifiers separately from the ensembles. So after

supplying each ensemble with the same base classifier, there could be no argue that some difference in the training methods of Troika and the other ensembles bias the results in favor of Troika; still one question remains. Is it possible that the selected binarization method (i.e. 1-1) in itself 'helps' Troika more than it 'helps' the other ensembles? To answer this question we needed to remake the experiments (at least some of them) using another kinds of binarization methods and find out if our primeval choice of binarization method is to blame with Troika's good performance.

The second reason was the disappointing results obtained by StackingC. StackingC is designed as an improvement of Stacking, and, as shown in our experiments, it had performed even worse than Stacking, especially in multiclass datasets which it should performed better. We suspected that StackingC is inefficient when its base-classifiers are trained using 1-1 binarization method; as there is greater number of base classifiers, the chance that each StackingC's meta classifier will predict correctly decreases and therefore StackingC's ability to correctly predict also decrease. Our experiments results emphasize this drawback; we see it baldly in *Segment, LED7, Vowel, KRKOPT and Letter* datasets. Each one of those datasets has at least 7 classes. There is a considerable drop of accuracy on those datasets compare to Stacking or Troika. Thus our hypothesis is that the performance of StackingC will increase dramatically, especially when using a multiclass datasets, when base classifiers binarization method will be changed.

5.3.1. Second experiment setup

The configuration of Troika, Stacking and StackingC were left untouched from previous experiment. We had, again, tested all ensembles using one, three and six induces. The difference, then, from first experiment, part from fewer tested datasets, is solely the method of binarization we used in the training of the base-classifiers.

5.3.2. Base-Classifiers Arrangements

We had tested three arrangements for base-classifiers trainings. Two are binarization methods; the 1-1 (one-against-one) and the 1-All (One-against-All) methods, and the last is a non binarization method AAA (All-against-All), in which

base classifiers train on the entire train-set, without any class-binarization (the default training method).

One-against-all (1AA) is the simplest arrangement method. When given a problem with *k* classes, *k* binary classifiers will be produced by using this method. Each classifier is exercised to distinguish a class C_i from the remaining classes. The final prediction is usually given by the classifier with the highest output value, as done in StackingC, or by combining them in some manner.

On the other hand we term the methodology used in the first experiment as one-against-one (1A1). Recall that this methodology consists $\frac{k(k-1)}{2}$ predictors, each differentiating a pair of classes C_i and C_j, where $i \neq j$. To combine the outputs produced by these classifiers, a majority voting scheme is applied. Each 1A1 classifier gives one vote to its preferred class. The final result is the class with most of the votes.

5.3.3. Second experiment results

Table 13 summarizes the average AUC results of all the ensemble methods; it shows that Troika performs better than its rivals regardless of binarization method. In total, it wins 7 out of the 9 (78%) experiments, 6 of 6 (100%) when using three or six inducers. The class binarization method has effect on the AUC; in average 1A1 yielded better AUC than the rest and 1AA yielded the worst. StackingC had won Stacking in 5 out of 6 experiments where base-classifiers training methods were 1-All and AAA. The evidence shows that troika is superior to Stacking and StackingC when using more than one inducer for base-classifiers.

.

Binarization method	#Inducers	Stacking	StackingC	Troika	Arrangement average
1A1	1	0.87	0.87	0.87	0.88
	3	0.88	0.9	0.91	
	6	0.86	0.86	0.88	
1AA	1	0.84	0.86	0.87	0.86
	3	0.86	0.85	0.89	
	6	0.85	0.88	0.89	
AAA	1	0.86	0.87	0.86	0.87
	3	0.85	0.88	0.89	
	6	0.85	0.88	0.89	
Average	-	0.86	0.87	0.88	

Table 13: Mean AUC (area under ROC) for all assembly methods

We can see from Table 12 that Troika performs better than its opponents, again, regardless of binarization method. In total, it wins 8 out of the 9 (89%) experiments, and 6 of 6 (100%) when given three or six inducers. AAA training method this time won second place. We got best performance using 1A1 coupled with three inducers. In a matter of fact, all ensemble methods had had their peak ROC results when using this exact configuration. The poorest results always came along with the used of single inducer regardless of base-classifiers training method, but, on the other hand, there wasn't hard evidence that using six inducers, rather than three, yields better results.

Binarization method	#Inducers	Stacking	StackingC	Troika	Arrangement average
1A1	1	3.27	1.42	1.66	44.73
	3	125.29	64.91	25.13	
	6	99.02	53.33	28.56	
1AA	1	4.59	1.41	2.3	6.03
	3	5.62	4.24	7.68	
	6	8.01	7.88	12.52	
AAA	1	4.69	0.34	0.68	14.14
	3	11.42	7.98	29.5	
	6	18.88	11.96	41.85	
Average	-	31.2	17.05	16.65	

Table 14: Mean execution time for all assembly methods.

We can see from Table 14 that Troika execution time is significantly better than its opponents when using 1-1 binarization method. One of the explanations for this particular result is that when using 1-1, base-classifiers count is much higher than when using 1AA, which is even higher than when using AAA. This proves our claim that when number of base-classifiers becomes greater Troika execution time, relatively to Stacking and StackingC, become longer in slower rate. We can see that Stacking took longer to execute when it used three inducers rather than using six. It is a very strange finding indeed, not quite comprehensible. Well, this phenomenal is a result of our distributed experiment. We had used different computers to perform experiments in which we used 1, 3 and 6 inducers. The one which run the experiments that used three inducers, happen to be a much slower computer, slow enough to make these ridiculous results. As a matter of fact, this does not affect our conclusions, because when we compare execution time of the three ensembles, we actually compare how each function, given the number of inducers and method of base-classifiers training, thus eliminating the between treatment effect.

Finally, we can see that Troika execution time was the longer then other ensembles when using AAA base-classifiers training method. In this particular method, where minimal base classifiers where induced, the large number of combining classifiers in Troika was the giving factor, therefore Troika execution time was the worst.

5.3.4. Summation of statistical results

So far the results have shown that troika excels Stacking and StackingC in terms of accuracy, regardless of base-classifiers binarization method. We also had shown Troika is preferable, in terms of execution time, especially when having many inducers to combine. In addition Troika's AUC mean is greater than of Stacking and of StackingC. However the statistical significance has been separately examined for each dataset. In this section we use the statistical procedure proposed in Demsar (2006) to conclude which algorithm performs best over multiple datasets. First we used adjusted non-parametric Friedman test with a confidence level of 95% to determine whether the difference between the ensemble methods is significance in

general. Later, if we found a significance difference, we used the Bonferroni-Dunn test with a confidence level of 95% (when not specified otherwise) to find which ensemble differs from Troika. The summary of the results are provided in Table 16. The "+" sign indicates that the degree of accuracy of Troika was significantly higher than the corresponding algorithm at a confidence level of 95%. The "=" sign indicates the accuracy was not significantly different. The "+ (xx%)" superscript indicates that the degree of accuracy of Troika was significantly higher than the corresponding algorithm at a confidence level of xx%. The summary table indicates that in most configurations, Troika has significantly prevailed Stacking and StackingC accuracy.

Table 15: Shows a summary of statistical significance of the difference between accuracy performances of the three ensemble methods.

Binarization method	Number of inducers	General Significance	Vs. Stacking	Vs StackingC
1A1	1	+	=	+
	3	+	+	+
	6	+	+	+
1AA	1	+	+	+ (91%)
	3	None	n/a	n/a
	6	+	+	+
AAA	1	None	n/a	n/a
	3	+	+	=
	6	+	+	=

A close examination of the results shown in Tables 10, 11 and 12 indicate that there are some datasets which disagree with our intermediate conclusions. Troika performs worse than Stacking in both Zoo and Soybea datasets. These poor performance of Troika in both datasets emphasize Troika's weaknesses. Zoo dataset has seven classes, 101 instances and when trained using 1-1 binarization method, 252 base-classifiers are trained. Soybean dataset contains 19 classes and 683 instances. We explain these poor results of Troika with the small number of instances in respect with number of classes. While Stacking and StackingC has one layer of combining layers, Troika has three. This attribute of Troika, forces Troika to spread the training meta instances with the three combining layers, therefore each layer gets fewer instances than Stacking or StackingC meta combiner gets. When there are enough instances in the datasets, this is not a major drawback. But when the original number of instances in the dataset is very small in respect to the number of classes, this may lead to an inferior ensemble. Specifically, we found the following index useful for

deciding which ensemble method to use. The index is defined as the number of instances divided by the square of the number of classes. For example in 1A1 binarization method Troika should be chosen if the dataset index is higher than 3.

Another important finding from our experimental study indicates that using StackingC along with base-classifiers which were trained using 1-1 binarization method yields very poor ensemble. This emphasizes another good attribute of Troika; assuming that some or all of the base-classifiers may not, or could not be trained on more than two classes of a multiclass datasets, then Troika will have a large advantage over StackingC and Stacking; over the first, for it yields poor ensemble when coupled with base-classifiers trained using 1-1 binarization method as already shown, and over the later, because it is simply not that good with multiclass datasets.

5.4. Comparing Troika to non Stacking ensembles methods

In the previous experiments we had tested Troika and compared it to other stacking generalization ensemble method such as Stacking and StackingC. It makes sense to first check whether Troika is capable of improving its closest relative ensembles. Now, however, we will compare Troika to other combing methods - Performance Weighting [Opitz and Shavlik, 1996], Distribution Summation [Clark and Boswell, 1991], Bayesian Combination [Buntine, 1990] and Naïve Bayes idea for combining various classifiers [John and Langley, 1995].

We used the same UCI dataset. We did not use binarization process on the base classifiers. We had used three induces – J48, VFI and IB1. From each inducer we had created two models using the 125% instances process that was explained earlier.

Ensemble Method	Bayesian Combination		Distribution Summation		Naïve Bayes Combination		Performance Weighting		Troika	
Dataset	**Accuracy**	**Rank**	**Accuracy**	**Rank**	**Accuracy**	**Rank**	**Accuracy**	**Rank**	**Accuracy**	**Rank**
Autos	79.64±8.09	2	80.17±6.74	1	79.10±9.76	3	75.71±5.60	5	76.64±9.56	4
Balance-scale	81.44±3.93	2.5	81.44±3.93	2.5	75.37±4.95	5	81.12±3.96	4	88.00±2.05	1
Breast-cancer	74.56±11.27	3	74.91±11.47	1.5	65.07±8.84	5	74.91±11.47	1	71.34±6.17	4
C-heart-disease	77.53±5.22	2.5	77.53±5.22	2.5	74.26±8.50	5	77.19±5.61	4	78.81±7.04	1
Credit-rating	84.06±3.74	2	83.19±4.59	3	78.26±3.35	5	81.30±4.90	4	86.96±3.55	1
Diabetes	71.45±5.56	2	70.93±5.20	4	67.81±4.32	5	71.19±5.22	3	73.41±4.65	1
Flag	58.95±11.32	4	59.47±11.65	2	55.79±12.21	5	58.95±10.76	3	59.47±12.90	1
German credit	72.20±3.01	3	72.00±2.94	4	68.90±3.57	5	72.30±2.87	2	74.90±3.84	1
Glass	71.81±8.41	1	70.86±10.09	3	68.33±7.95	5	71.36±10.95	2	69.83±8.63	4
Heart-statlog	75.56±8.59	2	75.56±8.94	3	74.81±8.34	5	75.19±8.56	4	75.93±8.95	1
Hepatitis_Domain	66.58±12.48	3	67.21±12.06	2	65.92±11.58	4	67.87±11.80	1	65.83±10.13	5
Horse-colic	82.09±6.83	2	81.82±7.16	3	79.90±7.65	5	81.54±6.30	4	85.57±5.06	1
House-votes-84	95.19±4.07	3	94.96±3.83	4	95.42±3.39	2	94.05±4.04	5	96.56±2.68	1
H-heart-disease	78.55±10.28	3.5	78.55±10.28	3.5	75.84±9.77	5	79.93±8.30	2	82.02±6.28	1
Ionosphere	91.75±3.06	2.5	91.75±3.06	2.5	87.48±6.45	4	87.19±4.67	5	92.89±3.32	1
Iris	96.67±4.71	2	96.67±4.71	2	94.67±6.13	4.5	96.67±4.71	2	94.67±5.26	4.5
Kr-vs-kp	99.31±0.41	3	99.25±0.37	4	99.44±0.48	1	98.28±1.00	5	99.37±0.47	2
Labor	89.67±11.91	1.5	89.67±11.91	1.5	82.67±11.20	5	88.33±15.81	3	84.33±12.48	4
Segment	97.32±0.84	4	97.36±0.80	2	96.62±0.61	5	97.36±0.88	1	97.32±0.76	3
Sonar	87.52±7.94	1	82.67±7.61	3	68.71±8.34	5	78.36±5.64	4	86.07±6.88	2
Soybean	92.38±2.49	3	91.94±2.81	4	92.39±2.16	2	92.82±2.02	1	91.06±3.07	5
Splice	94.45±2.0	2	90.38±1.75	4	91.1±2.04	3	80.16±1.8	5	95.99±1.6	1
Vehicle	73.88±4.57	2	72.58±4.56	4	71.17±5.52	3	71.52±4.37	5	73.76±4.17	1
W-beast-cancer	95.99±2.85	3.5	95.99±2.85	3.5	95.42±3.08	5	96.85±2.69	1	96.13±2.35	2
Zoo	98.89±3.51	2.5	98.89±3.51	2.5	99.00±3.16	1	97.89±4.46	4	96.78±7.38	5
Average Rank	**2.46**		**2.84**		**4.18**		**3.18**		**2.34**	

Table16: Comparing the prediction accuracy of several combining ensemble methods to Troika on 24 UCI datasets. In this experiment we had used three inducers: J48, VFI and IBk. We used each induces to produces two models so each combining method had combined six classification models. Please notice that each result contain three data items; the predictive accuracy, standard deviation and rank. For example the value "79.64±8.09, 2" indicate of accuracy of 79.64%, standard deviation of 8.09 and rank of 2. Smaller rank is better.

Table 16 shows that among all five tested combining methods Troika had achieved the smallest average rank. The second best is Bayesian combination, then

Distribution summation followed by performance weighting. The worst combining method was found to be the Naïve Bayes combination method. The non-parametric Freidman test result had shown that at least one combining method performance is statistically significantly different. Despite of the nice rank figures, we found out using the Bonferroni-Dunn test that Troika is only statistically better than Naïve Bayes combination method. The difference between the other combining methods was statistically insignificant (using α=0.05).

One of the several noticeable disadvantages of using Meta learner as combining method, such as Stacking, StackingC and mostly Troika, is that it takes extra time to train the Meta learner. This fact was very evident in this experiment. No statistical test was needed to be done; the time it had taken to train Troika was more than ten time slower that the other combining methods. However, when training time is not very important than the better accuracy of Troika makes it the preferable combining method.

5.5. How Troika affected from dataset's class count?

In this section we investigate another interesting parameter; the effect of classes number on Troika performance. In order to answer the question in the title we took the KRKOPT dataset (instances distribution shown in table 17), which initially has 17 classes and manipulated it several times. This dataset has been examined with increasing number of classes in order to examine the relation between the number of classes and the predictive performance. The manipulation was very simple; for the creation of the first derived dataset, "KRKOPT-2-clss" we started with the original KRKOPT dataset and filter only instances of the two most prominent classes (the classes which has the most instances), then to create the "KRKOPT-3-clss" we did exactly the same procedure as with "KRKOPT-2-clss", but filtered only the instances of the three most prominent classes and so on with "KRKOPT-4-clss", "KRKOPT-5-clss", "KRKOPT-6-clss" and "KRKOPT-7-clss". At the end of this process we had six datasets; each has different class count and different number of instances. Table 18 specifies the list of datasets which were used in this experiment.

Class label	Instances
fourteen	4553
thirteen	4194
twelve	3597
eleven	2854
Draw	2796
fifteen	2166
ten	1985
nine	1712
eight	1433
seven	683
six	592
five	471
sixteen	390
Two	246
four	198
Three	81
One	78
Zero	27

Table 17: The original KRKOPT dataset's instances distribution ordered by instances quantity.

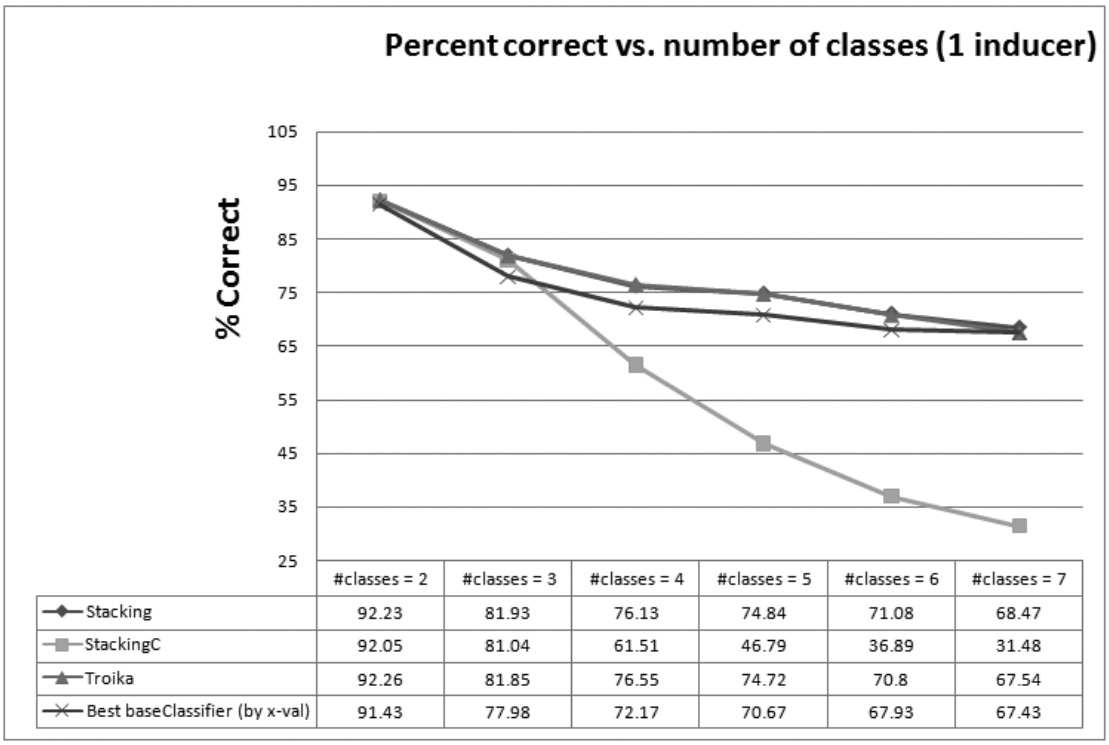
Dataset name	Contained classes
KRKOPT-2clss	Eleven, Draw
KRKOPT-3clss	Eleven, Draw, Fifteen
KRKOPT-4clss	Eleven, Draw, Fifteen, Ten
KRKOPT-5clss	Eleven, Draw, Fifteen, Ten, Nine
KRKOPT-6clss	Eleven, Draw, Fifteen, Ten, Nine, Eight
KRKOPT-7clss	Eleven, Draw, Fifteen, Ten, Nine, Eight, Seven

Table 18: Six KRKOPT derived datasets

5.5.1. Experiment results:

In this experiment we use the same ensembles configuration as we did in the previous two experiments. Our metrics had not been changed; accuracy, AUC and execution time. All base-classifiers where trained using 1-1 binarization method.

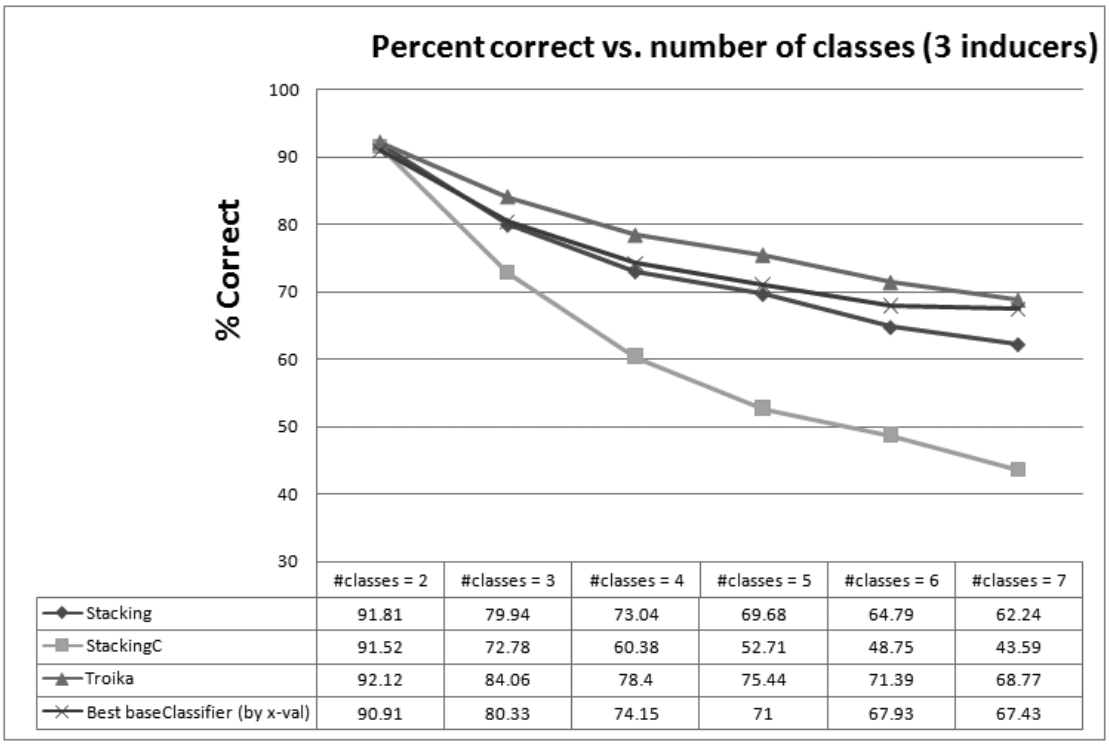
Figures 7, 8 and 9 present the accuracy results of Troika, Stacking and StackingC when using 1, 3 and 6 inducers respectively. We show here only the Accuracy



	#classes = 2	#classes = 3	#classes = 4	#classes = 5	#classes = 6	#classes = 7
Stacking	92.23	81.93	76.13	74.84	71.08	68.47
StackingC	92.05	81.04	61.51	46.79	36.89	31.48
Troika	92.26	81.85	76.55	74.72	70.8	67.54
Best baseClassifier (by x-val)	91.43	77.98	72.17	70.67	67.93	67.43

Figure 7: showing the correct prediction rate as function of classes count in datasets using one inducer (J48).

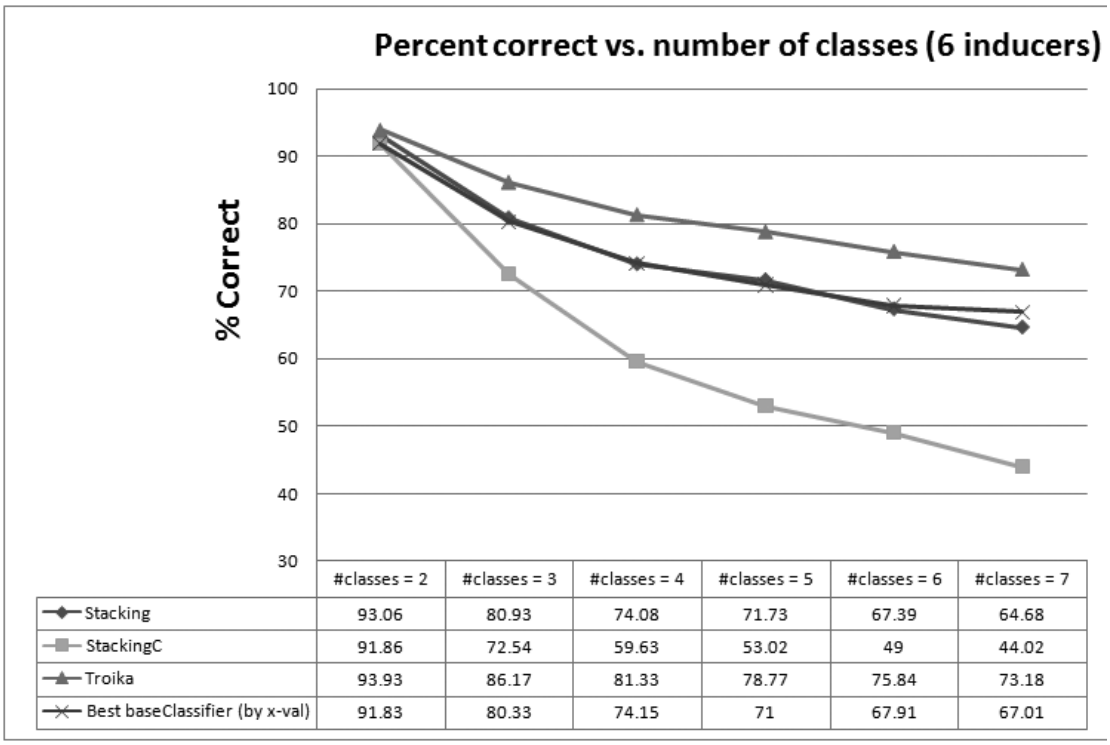
From the Figure 7, it is evident that Troika and Stacking are pretty much close with small advantage to stacking. This came with no surprise to us since we were using only one inducer. On the other hand, StackingC accuracy performance is free falling while number of classes in datasets increases. This is the effect we had already seen in previous experiments. Again the blame is the 1-1 binarization method we used to train the base-classifier. Stacking and Troika accuracy is better than the best base-classifier selected by cross-validation.



	#classes = 2	#classes = 3	#classes = 4	#classes = 5	#classes = 6	#classes = 7
Stacking	91.81	79.94	73.04	69.68	64.79	62.24
StackingC	91.52	72.78	60.38	52.71	48.75	43.59
Troika	92.12	84.06	78.4	75.44	71.39	68.77
Best baseClassifier (by x-val)	90.91	80.33	74.15	71	67.93	67.43

Figure 8: showing the correct prediction rate as function of classes count in datasets using three inducers (J48, IBK, and VFI)

From Figure 8, we can learn that Stacking and Troika have same accuracy rate when having 2-classes dataset. Then, when number of classes is three a gap between the accuracy of those two is formed, in favor of Troika. As the number of classes count in data-set increases, this gap enlarges. StackingC accuracy performance continues to free falling while number of classes in datasets increases. This time, Troika alone has better accuracy than the best base-classifier selected by cross-validation. It seems that adding inducers to ensemble had damaged Stacking accuracy.



	#classes = 2	#classes = 3	#classes = 4	#classes = 5	#classes = 6	#classes = 7
Stacking	93.06	80.93	74.08	71.73	67.39	64.68
StackingC	91.86	72.54	59.63	53.02	49	44.02
Troika	93.93	86.17	81.33	78.77	75.84	73.18
Best baseClassifier (by x-val)	91.83	80.33	74.15	71	67.91	67.01

Figure 9: the correct prediction ratio as function of classes count in datasets using six inducers (J48, IBK, VFI, Part, BayesNet and SMO)

From Figure 9, we can learn that Stacking and Troika have approximately the same accuracy rate when having 2-classes dataset. Then, when number of classes is three a gap between the accuracy of those two is formed, in favor of Troika. As the number of classes count in data-set increases, this gap enlarges exactly as it was when used three inducers. This time, the addition of three more inducers helped both Troika and Stacking. As much as this addition aided Stacking, Troika was still the solely ensemble which had better accuracy than the best base-classifier selected by cross-validation. It seems that adding inducers to ensemble did not help to StackingC accuracy at all, as was expected.

To conclude the above results, Troika produce better predictions, especially when dealing with multiclass datasets; the greater number of classes, the larger the gap becomes between Troika and the other ensemble methods.

6. Conclusions and Future Work

In this paper we have presented a new ensemble method, Troika. It is an improvement of Stacking ensemble scheme. While Stacking uses one meta classifier in a rather simplistic manner, Troika, has three layers of combining classifiers.

The goal of Troika was to address Stacking weaknesses, namely, it weak performance with multiclass datasets, and it had done it nicely. It had done that while preserving execution time as or better than of Stacking's when many base-classifiers participated in the ensemble.

The even greater challenge we face in this work was to make Troika better than StackingC, which was targeting the same objectives and had the advantage of already having proofs of being better than Stacking. We were somewhat amused to see StackingC performed worse than Stacking and Troika in most of our first experiment. Later, in the second experiment it regained his title, as a superior to Stacking, but remained inferior to Troika in terms of its prediction accuracy.

We had performed three major experiments in this work. First one was aimed to evaluate Troika performance over a large group of dataset. We depended solely on 1-1 binarization method for base-classifiers training, and made use of one, three and six induces in order to find whether the number of inducers affect Troika performance. Second major experiment goal was to show if Troika good results, obtained in the first experiment, are merely base-classifiers training method depended. Our third and last major experiment was dealing with the question of the ability of Troika to preserve its being best ensemble method while number of classes in dataset increases. For this experiment we have devised six unique datasets, which helped us answer this hanging question.

From all the experiment we had learned that troika produces better results than Stacking and StackingC in terms of accuracy, no matter which base-classifiers binarization method we had chosen, furthermore, we have found that when execution time is not critical, Troika is one of the best choices among non stacking combining

methods as well. Regarding the execution time, our experiments had indicated that Troika is preferable only when the base classifiers are trained using the 1A1 binarization method. Troika showed better AUC mean than of Stacking and of StackingC, even though we couldn't get the needed statistical support with high confidence level due to high variance among the results. We also showed one Troika's drawbacks which are not negligible.

Additional issues to be further studied include:

- Optimize Troika's combining classifiers – in this work we used only one kind of inducer for all combining classifiers. Our intuition tells us that a progress may be done in this direction. Find out which inducer fits which combiner layer.
- Optimize Troika training process – in Troika we assumed inner cross-validation process in training phase of 5x5, meaning five cross validation for specialist and five for Meta classifiers (Super classifier wasn't trained using cross validation process). Whether this decision was optimal, we do not know.
- Train Meta Classifier specifically to be pattern sensitive. We have shown that Troika Meta classifiers job comes down to recognition of one of three possible patterns. We would like to have the Meta classifiers address this problem more efficiently.

References

1. Anand R, Methrotra K, Mohan CK, Ranka S. Efficient classification for multiclass problems using modular neural networks. IEEE Trans Neural Networks, 6(1): 117-125, 1995.
2. Aydın U, Murat S, Olcay T Yıldız, Ethem A. Incremental construction of classifier and discriminant ensembles. Information science, Volume 179, Issue 9, 15 April 2009, Pages 1298-1318
3. Bauer, E. and Kohavi, R., An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants". Machine Learning, 35: 1-38, 1999.
4. Boser, B.E., Guyon, I.M. & Vapnik, V.N. (1992). A training algorithmfor optimal margin classifiers. 5th Annual ACM (pp. 144–152). ACM Press, Pittsburgh, PA.
5. Buntine, W., 1990. A Theory of Learning Classification Rules. Doctoral dissertation. School of computing Science, University of Technology, Sydney.
6. Breiman, L. (1996). Bagging predictors. Machine Learning , 123-140.
7. Clark, P., and Boswell, R., 1991. Rule Induction with CN2: Some recent improvements. Proc. of the European Working Session on Learning, pp. 151-163.
8. Cooper, G. F. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. Machine Learning (9) , 309-347.

9. Demsar J., Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
10. Dietterich, T. G., and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research, 2:263-286, 1995.
11. Domingos, P., Using Partitioning to Speed Up Specific-to-General Rule Induction. In Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models, pp. 29-34, AAAI Press, 1996.
12. Dzeroski S. and Zenko B. (2002). Stacking with Multi-Response Model Trees. Multiple Classifier Systems , 201-211 .
13. Dzeroski S., Zenko B. . (2004). Is Combining Classifiers with Stacking Better than Selecting the Best One? Machine Learning 54(3), (pp. 255-273).
14. F. Provost and T. Fawcett (2001). Robust classification for imprecise environments. Machine Learning, 42, 203–231.
15. Fürnkranz, J. (2002). Pairwise Classification as an Ensemble Technique. European Conference on Machine Learning (pp. 97-110). Helsinki, Finland: Austrian Research Institute for Artificial Intelligence.
16. G. Peter Zhang, A neural network ensemble method with jittered training data for time series forecasting, Information Sciences: an International Journal, v.177 n.23, p.5329-5346, December, 2007
17. Geman S., Bienenstock, E., and Doursat, R., Neural networks and the bias variance dilemma. Neural Computation, 4:1-58, 1995.
18. Hastie T., R. Tibshirani, Classification by pairwise coupling, The Annals of Statistics 2 (1998) 451–471.
19. John, G. H., and Langley, P., 1995. Estimating Continuous Distributions in Bayesian Classifiers. In Proc. of the Conference on Uncertainty in Artificial Intelligence, pp. 338-345
20. Knerr S., L. Personnaz, G. Dreyfus, Single-layer learning revisited: a stepwise procedure for building and training a neural network, Springer-Verlag, 1990, pp.41–50.
21. Kreβel U., Pairwise classification and support vector machines, in: B. Schölkopf, C. J. C. Burges, A. J. Smola (Eds.), Advances in Kernel Methods – Support Vector Learning, MIT Press, 1999, pp. 185–208.
22. Lorena A. and de Carvalho A. C. P. L. F. : Evolutionary Design of Code-matrices for Multiclass Problems, Soft Computing for Knowledge Discovery and Data Mining, Springer US ,153-184, 2007
23. Lu B.L., Ito M., Task Decomposition and Module Combination Based on Class Relations: A Modular Neural Network for Pattern Classification, IEEE Trans. on Neural Networks, 10(5):1244-1256, 1999.
24. Merz C. J, and Murphy P.M., UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science, 1998.
25. Merz, C. (1999), 'Using correspondence analysis to combine classifiers', Machine Learning 36, 33–58
26. Mitchell, T., Machine Learning, McGraw-Hill, 1997.
27. Opitz, D. and Maclin, R., Popular Ensemble Methods: An Empirical Study, Journal of Artificial Research, 11: 169-198, 1999.
28. Quinlan, J. R., Bagging, Boosting, and C4.5. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pages 725-730, 1996.
29. Quinlan, R. (1993). C4.5: Programs for Machine Learning. Machine Learning , 235-240.
30. S. Cohen, L. Rokach, O. Maimon (2007), Decision Tree Instance Space Decomposition with Grouped Gain-Ratio, Information Science, 177(17):3592-3612.
31. Seewald A.K. and J. Fuernkranz. (2001). An Evaluation of Grading Classifiers. Advances in Intelligent Data Analysis: 4th International Conference (pp. 115-124). Berlin/Heidelberg/New York/Tokyo: Springer.
32. Seewald A.K.. (2003). Towards understanding stacking - Studies of a general ensemble learning scheme. PhD-Thesis, TU Wien.
33. Seewald, A. (2002). How to Make Stacking Better and Faster While Also Taking Care of an Unknown Weakness. Nineteenth International Conference on Machine Learning (pp. 554-561). sydney: Morgan Kaufmann Publishers.

34. Sivalingam D., Pandian N., Ben-Arie J., Minimal Classification Method With Error-Correcting Codes For Multiclass Recognition, International Journal of Pattern Recognition and Artificial Intelligence 19(5): 663 - 680, 2005.
35. Ting, K. M., Witten, I. H. (1999): Issues in stacked generalization. Journal of Artificial Intelligence Research 10, pages 271-289.
36. Opitz D. and Shavlik, J., 1996. Generating Accurate and Diverse Members of a Neural Network Ensemble. Advances in Neural Information Processing Systems, vol. 8, pp. 535-541.
37. Witten I. H. and Frank E. (2005) "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
38. Wolpert, D. (1992). Stacked Generalization. Neural Networks5 , 241-259.Boser,BE., Guyon,l.M. and Vapnik,VN. (1992) A training algorithm for optimal margin classifiers. In Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory ACM Press, Pittsburgh, PA, pp. 144-152.
39. Zhoua J., Pengb H., Suenc C., Data-driven decomposition for multi-class classification, Pattern Recognition 41: 67 – 76, 2008.
40. Zenobi, G., and Cunningham, P. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In Proceedings of the European Conference on Machine Learning, 2001.